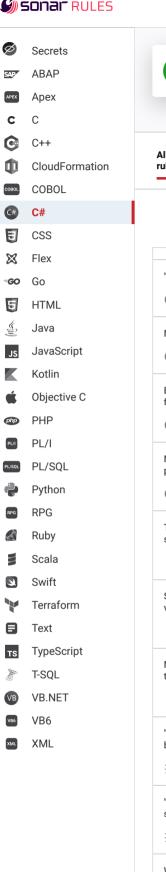
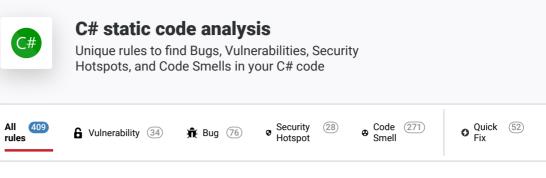
Q

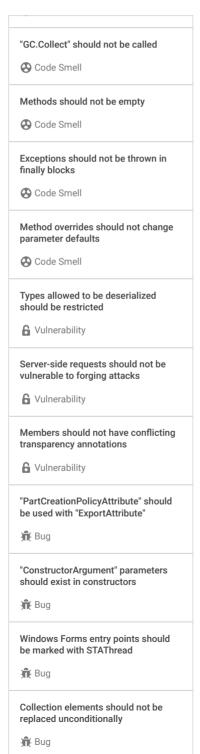
Search by name.







Tags



Exceptions should not be created

without being thrown

```
"Shared" parts should not be
                                              Analyze your code
created with "new"
Rug Oritical O
                          mef pitfall
Marking \ a \ class \ with \ {\tt PartCreationPolicy(CreationPolicy.Shared), which}
is part of Managed Extensibility Framework (MEF), means that a single, shared
instance of the exported object will be created. Therefore it doesn't make sense to
create new instances using the constructor and it will most likely result in
unexpected behaviours.
This rule raises an issue when a constructor of a class marked shared with a
PartCreationPolicyAttribute is invoked.
Noncompliant Code Example
  [Export(typeof(IFooBar))]
 [PartCreationPolicy(CreationPolicy.Shared)]
 public class FooBar : IFooBar
 {
 public class Program
      public static void Main()
           var fooBar = new FooBar(); // Noncompliant;
  }
Compliant Solution
 [Export(typeof(IFooBar))]
  [PartCreationPolicy(CreationPolicy.Shared)]
 public class FooBar : IFooBar
 public class Program
      public static void Main()
      {
           var fooBar = serviceProvider.GetService<IFooBar>();
 Available In:
 sonarlint ⊕ | sonarcloud ♦ | sonarqube
```

Collection sizes and array length comparisons should make sense

Bug

Serialization event handlers should be implemented correctly

Bug

Descrialization methods should be

provided for "OptionalField" members

All branches in a conditional structure should not have exactly the same

👬 Bug

© 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