








-  **sonar** RULES
-  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



Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

All rules 632

 Vulnerability 53

 Bug 154

 Security Hotspot 36

 Code Smell 389

 Quick Fix 42

Tags ▾

Search by name... 🔍

Abstract class names should comply with a naming convention

 Code Smell

Strings literals should be placed on the left side when checking for equality

 Code Smell

Files should contain an empty newline at the end

 Code Smell

Source code should be indented consistently

 Code Smell

A close curly brace should be located at the beginning of a line

 Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines

 Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line

 Code Smell

An open curly brace should be located at the beginning of a line

 Code Smell

An open curly brace should be located at the end of a line

 Code Smell

Tabulation characters should not be used

 Code Smell

Functions should not be defined with a variable number of arguments

 Code Smell

Spring "@Controller" classes should not use "@Scope"

Analyze your code

 Bug  Major   spring

Spring @Controllers, @Services, and @Repositories have singleton scope by default, meaning only one instance of the class is ever instantiated in the application. Defining any other scope for one of these class types will result in needless churn as new instances are created and destroyed. In a busy web application, this could cause a significant amount of needless additional load on the server.

This rule raises an issue when the @Scope annotation is applied to a @Controller, @Service, or @Repository with any value but "singleton". @Scope( "singleton" ) is redundant, but ignored.

Noncompliant Code Example

```
@Scope("prototype") // Noncompliant
@Controller
public class HelloWorld {
```

Compliant Solution

```
@Controller
public class HelloWorld {
```

Available In:

 |  | 

© 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](#)

<div>Local-Variable Type Inference should be used</div> <div> Code Smell</div>
<div>Migrate your tests from JUnit4 to the new JUnit5 annotations</div> <div> Code Smell</div>
<div>Track uses of disallowed classes</div> <div> Code Smell</div>
<div>Track uses of "@SuppressWarnings" annotations</div> <div> Code Smell</div>