

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 rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

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 beans should be considered by "@ComponentScan"

Analyze your code

Code Smell

Critical

spring pitfall

Spring beans belonging to packages that are not included in a @ComponentScan configuration will not be accessible in the Spring Application Context. Therefore, it's likely to be a configuration mistake that will be detected by this rule.

**Note:** the @ComponentScan is implicit in the @SpringBootApplication annotation, case in which Spring Boot will auto scan for components in the package containing the Spring Boot main class and its sub-packages.

**Noncompliant Code Example**

```
package com.mycompany.app;

@Configuration
@ComponentScan("com.mycompany.app.beans")
public class Application {
    ...
}

package com.mycompany.app.web;

@Controller
public class MyController { // Noncompliant; MyController be
    ...
}
```

**Compliant Solution**





```
package com.mycompany.app;

@Configuration
@ComponentScan({"com.mycompany.app.beans", "com.mycompany.app
or
@ComponentScan("com.mycompany.app")
or
@ComponentScan
public class Application {
    ...
}

package com.mycompany.app.web;

@Controller
public class MyController { // "com.mycompany.app.web" is re
    ...
}
```

Available In:

<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>

sonarlint

sonarcloud

sonarqube

© 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](#)