




 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

"@EnableAutoConfiguration" should be fine-tuned

Analyze your code

Code Smell

Major ?

"@EnableAutoConfiguration" is a convenient feature to configure the Spring Application Context by attempting to guess the beans that you are likely to need. The drawback is that it may load and configure beans the application will never use and therefore consume more CPU and RAM than really required. @EnableAutoConfiguration should be configured to exclude all the beans not required by the application. Alternatively, use the @Import annotation instead of @EnableAutoConfiguration, to explicitly import the useful AutoConfiguration classes.

This rule applies for @SpringBootApplication as well.

Noncompliant Code Example

```
@SpringBootApplication
public class MyApplication {
    ...
}
```

Compliant Solution





```
@SpringBootApplication(exclude = {
    MultipartAutoConfiguration.class,
    JmxAutoConfiguration.class,
})
public class MyApplication {
    ...
}
```

```
@Configuration
@EnableAutoConfiguration(exclude = {
    MultipartAutoConfiguration.class,
    JmxAutoConfiguration.class,
})
public class MyApplication {
    ...
}
```

```
@Configuration
@Import({
    DispatcherServletAutoConfiguration.class,
```

https://rules.sonarsource.com/java/RSPEC-4604

1/2

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

```
EmbeddedServletContainerAutoConfiguration.class,  
ErrorMvcAutoConfiguration.class,  
HttpEncodingAutoConfiguration.class,  
HttpMessageConvertersAutoConfiguration.class,  
JacksonAutoConfiguration.class,  
ServerPropertiesAutoConfiguration.class,  
PropertyPlaceholderAutoConfiguration.class,  
ThymeleafAutoConfiguration.class,  
WebMvcAutoConfiguration.class  
  
})  
public class MyApplication {  
  ...  
}
```

Deprecated

This rule is deprecated, and will eventually be removed.

Available In:
 |  | 