
































-  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




# PHP static code analysis

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


All rules 268 Vulnerability 40 Bug 51 Security Hotspot 33 Code Smell 144

Tags ▾


Search by name... 

 Code Smell


More than one property should not be declared per statement

 Code Smell


The "var" keyword should not be used

 Code Smell


"<?php" and "<?=" tags should be used

 Code Smell


File names should comply with a naming convention

 Code Smell


Comments should not be located at the end of lines of code

 Code Smell

Local variable and function parameter names should comply with a naming convention

 Code Smell

Field names should comply with a naming convention

 Code Smell

Lines should not end with trailing whitespaces

 Code Smell

Files should contain an empty newline at the end

 Code Smell

Modifiers should be declared in the correct order

 Code Smell

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

 Code Smell

## A subclass should not be in the same "catch" clause as a parent class

Analyze your code

 Code Smell  Minor  unused

Repeating an exception class in a single catch clause will not fail but it is not what the developer intended. Either the class is not the one which should be caught, or this is dead code.

Having a subclass and a parent class in the same catch clause is also useless. It is enough to keep only the parent class.

This rule raises an issue when an exception class is duplicated in a catch clause, or when an exception class has a parent class in the same catch clause.

### Noncompliant Code Example

```
try {
    throw new CustomException();
} catch(CustomException | Exception $e) { // Noncompliant. C
    echo $e->message();
}

try {
    throw new CustomException();
} catch(Exception | Exception $e) { // Noncompliant.
    echo $e->message();
}
```

### Compliant Solution

```
try {
    throw new CustomException();
} catch(Exception $e) {
    echo $e->message();
}

try {
    throw new CustomException();
} catch(CustomException $e) {
    echo $e->getCustomMessage();
} catch(Exception $e) {
    echo $e->message();
}
```

### See

- RFC - [Catching Multiple Exception Types](#)

Available In:

sonarlint  sonarcloud  sonarqube 


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

 Code Smell


Tabulation characters should not be used

 Code Smell

Method and function names should comply with a naming convention

 Code Smell

Creating cookies with broadly defined "domain" flags is security-sensitive

 Security Hotspot

© 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](#)