Atitit pmd sumup code check static

目录

[1. 1.3、Java 静态代码分析理论基础和主要技术 1](#_Toc4291)

[1.1. 缺陷模式匹配： 1](#_Toc24362)

[1.2. 类型推断： 1](#_Toc8676)

[1.3. 模型检查： 2](#_Toc10073)

[1.4. 数据流分析： 2](#_Toc26561)

[2. Review code 2](#_Toc18208)

[3. PMD附带了许多可以直接使用的规则， 3](#_Toc20075)

[4. Pmd level 3](#_Toc29380)

[5. 重要规则 红色block 4](#_Toc6080)

[5.1. ReturnEmptyArrayRatherThanNull 4](#_Toc28840)

[5.2. Avoid Throwing RawException Types 4](#_Toc21643)

[6. Pmd rules 425 4](#_Toc19067)

[6.1. Rule setg 4](#_Toc31629)

[6.2. Priority level 5](#_Toc2212)

[7. Pmd best practis 5](#_Toc13741)

[7.1. Ajd name priority to wawring in perf>>pmd>rule cfg 5](#_Toc21016)

[8. Pmd tools 5](#_Toc28911)

[8.1. Datafolw view 5](#_Toc23571)

[8.2. Ast tool xpath designer 7](#_Toc6112)

# 1.3、Java 静态代码分析理论基础和主要技术

## 缺陷模式匹配：

缺陷模式匹配事先从代码分析经验中收集足够多的共性缺陷模式，将待分析代码与已有的共性缺陷模式进行模式匹配，从而完成软件的安全分析。这种方式的优点是简单方便，但是要求内置足够多缺陷模式，且容易产生误报。

## 类型推断：

类型推断技术是指通过对代码中运算对象类型进行推理，从而保证代码中每条语句都针对正确的类型执行。这种技术首先将预定义一套类型机制，包括类 型等价、类型包含等推理规则，而后基于这一规则进行推理计算。类型推断可以检查代码中的类型错误，简单，高效，适合代码缺陷的快速检测。

## 模型检查：

模型检验建立于有限状态自动机的概念基础之上，这一理论将被分析代码抽象为一个自动机系统，并且假设该系统是有限状态的、或者是可以通过抽象归 结为有限状态。模型检验过程中，首先将被分析代码中的每条语句产生的影响抽象为一个有限状态自动机的一个状态，而后通过分析有限状态机从而达到代码分析的 目的。模型检验主要适合检验程序并发等时序特性，但是对于数据值域数据类型等方面作用较弱。

## 数据流分析：

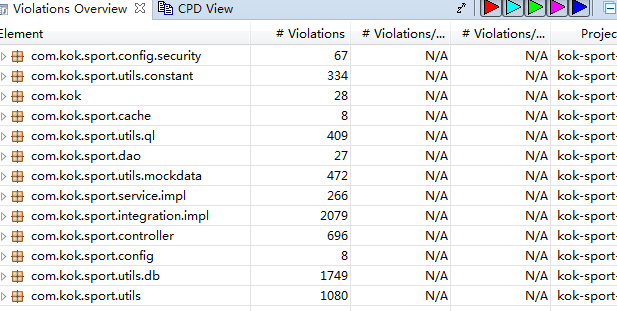
数据流分析也是一种软件验证技术，这种技术通过收集代码中引用到的变量信息，从而分析变量在程序中的赋值、引用以及传递等情况。对数据流进行 分析可以确定变量的定义以及在代码中被引用的情况，同时还能够检查代码数据流异常，如引用在前赋值在后、只赋值无引用等。数据流分析主要适合检验程序中的 数据域特性。

————————————————

版权声明：本文为CSDN博主「lfendo」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/u011781521/article/details/78834031

# Review code



# PMD附带了许多可以直接使用的规则，

利用这些规则可以找出Java源程序的许多问题，例如：

® 潜在的bug：空的try/catch/finally/switch语句

® 未使用的代码：未使用的局部变量、参数、私有方法等

® 可选的代码：String/StringBuffer的滥用

® 复杂的表达式：不必须的if语句、可以使用while循环完成的for循环

® 重复的代码：拷贝/粘贴代码意味着拷贝/粘贴bugs

® 循环体创建新对象：尽量不要再for或while循环体内实例化一个新对象

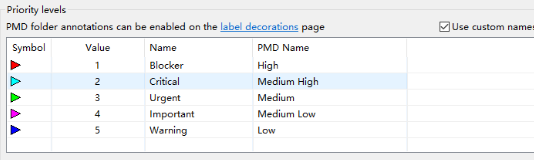
@ 资源关闭：Connect，Result，Statement等使用之后确保关闭掉

此外，用户还可以自己定义规则，检查Java代码是否符合某些特定的编码规范。例如，你可以编写一个规则，要求PMD找出所有创建Thread和Socket对象的操作。

# Pmd level

Blocker is high，critical med high，>>>warning is lowest

————————————————



# 重要规则 红色block

## ReturnEmptyArrayRatherThanNull

**private** **static** **char**[] genetmp(String string) {

String[] a = string.split(",");

**for** (String c : a) {

System.***out***.print("#{" + c + "},");

}

**return** **null**;

Avoid Reassigning Parameters

## Avoid Throwing RawException Types

**try** {

meth\_query = MybatisMapper.**class**.getMethod("query", Map.**class**);

} **catch** (NoSuchMethodException | SecurityException e) {

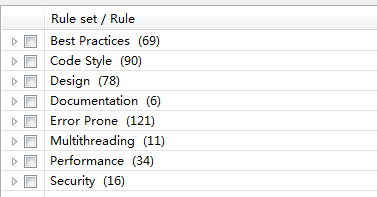
**throw** **new** RuntimeException(e);

}

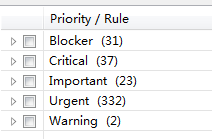
FormalParameterNamingConventio  
DataflowAnomaly Analysis  
UnusedImports  
SystemPrintln  
LocalVariableNaming Conventions  
VariableNaming Conventions

# Pmd rules 425

## Rule setg



## Priority level



## Priority / Rule Blocker (18)

AbstractClass Without AnyMethod  
AvoidFileStream  
Avoid Throwing NullPointerException  
Avoid Throwing RawExceptionTypes  
Avoid Trailing Comma  
Avoid UsingShortType  
Avoid WithStatement  
ClassWithOnlyPrivate ConstructorsShouldBeFinal  
ConstructorCallsOverridableMethod  
DoubleChecked Locking  
EmptyMethodInAbstractClassShouldBeAbstract  
EqualsNull  
GlobalVariable  
OneDeclaration Perline  
ReturnEmptyArrayRatherThanNull  
Scope ForInVariable  
UnreachableCode  
UseBaseWithParseInt

# Pmd best practis

## Ajd name priority to wawring in perf>>pmd>rule cfg

# Pmd tools

## Datafolw view

Atitit java code parse graph meth by pmd

**public** **static** String getValWarpFmt(String cType, Object object) {

**if** (cType.equals("BIGINT")) {

**return** object.toString();

}

**if** (cType.equals("TINYINT") || cType.equals("INT"))

**return** object.toString();

**if** (cType.equals("VARCHAR"))

**return** "'" + object.toString() + "'";

**if** (cType.equals("DATETIME")) {

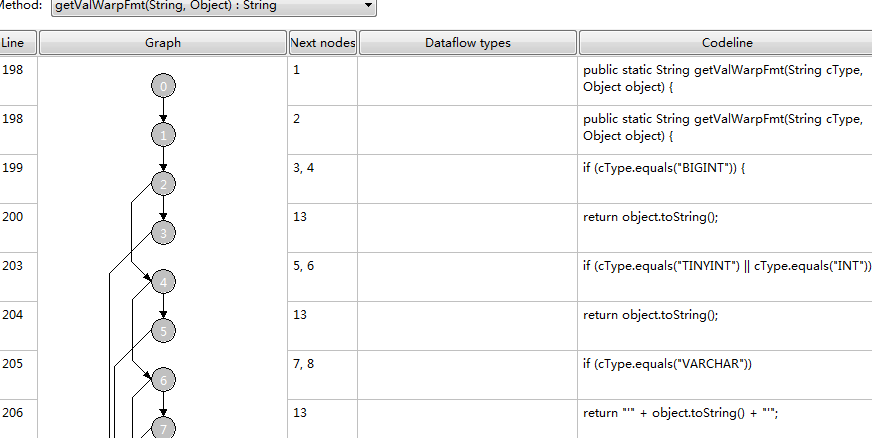
**if** (object **instanceof** Map)

**return** ((Map) object).get("exp").toString();

**else**

**return** "'" + object.toString() + "'";

}



## Ast tool xpath designer

