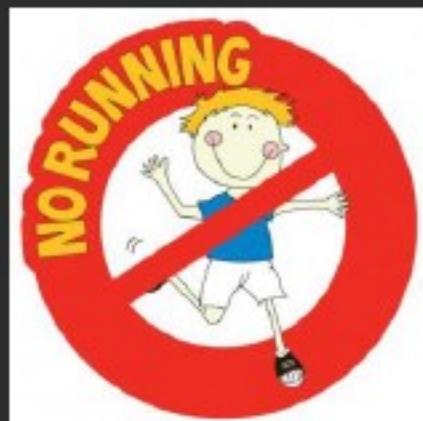


JAVA静态代码检查那些事

What : 什么是静态代码检查 ?

静态代码检查 - Static Analysis (SA)
Static Application Security Testing (SAST)



简单的说 - 在不运行代码的方式下发现代码中的 bugs

具体的说 - 在不运行代码的方式下，通过词法分析、语法分析、控制流、数据流分析等技术对程序代码 (source code/Binaries/Configuration files) 进行扫描，验证代码是否满足规范性、安全性、可靠性、可维护性等指标的一种代码分析技术

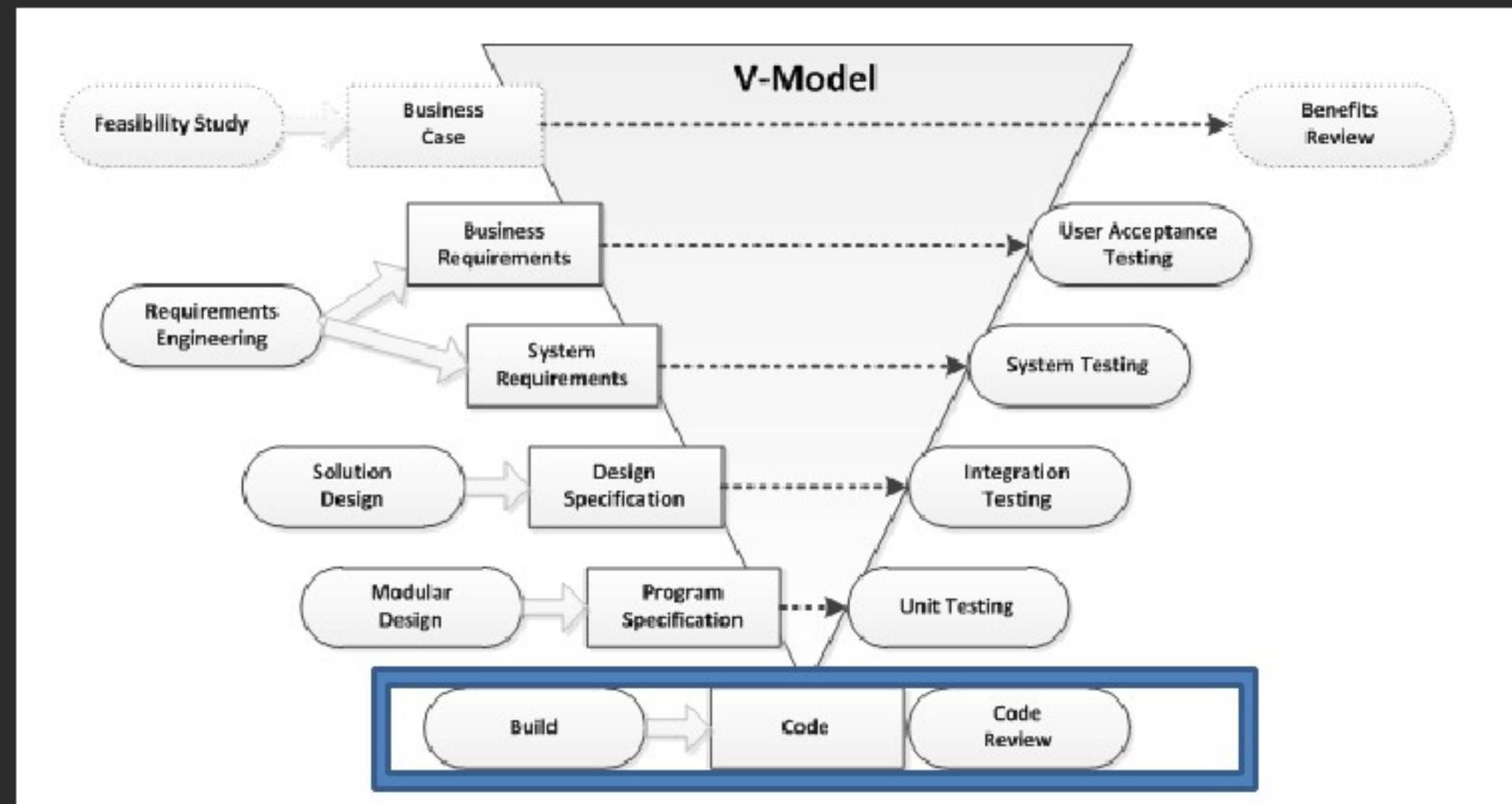
Why : 为什么使用静态代码检查 ?

先来看看什么是 “动态代码检查”

Dynamic analysis: 需要通过执行程序来发现应用内存泄漏、输入输出、性能的问题

如：单元测试

Why : 为什么使用静态代码检查 ?



Where and When : 使用静态代码检查 ?

开发过程中、完成时发现代码问题

- 本地IDE 插件----开发过程中
- Code review----提测前
- 持续集成---整个开发流程 (daily build)

HOW：怎么样做JAVA静态代码检查？

- JAVA静态代码检查常用工具
 - CheckStyle
 - PMD
 - Findbugs
 - 本地IDE配置及结果分析
- Sonar安装SA 插件及自定义配置
- 静态代码检查原理

静态代码检查常用工具-Checkstyle

特点：通过检查对代码编码格式，命名约定，Javadoc，类设计等方面进行代码规范和风格的检查

目标：实现代码风格的一致性

检查项：

- Javadoc注释
- 命名约定
- 标题
- Import语句
- 体积大小
- 空白
- 修饰符
- 块
- 代码问题
- 类设计
- 重复代码



静态代码检查常用工具-Checkstyle

实例：

```

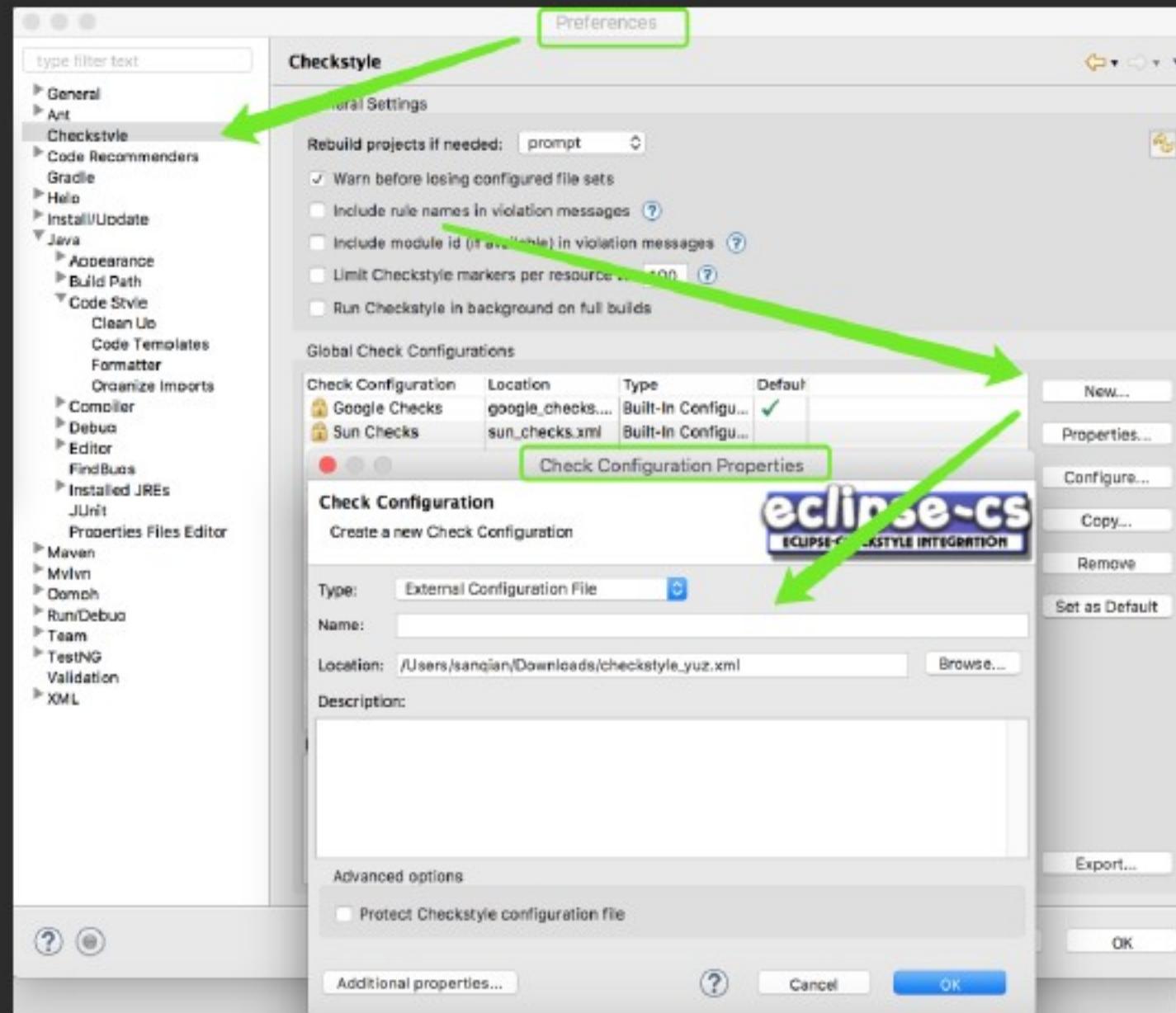
6 package com.netease.qa.cloud.nce.testcase.openapi.microservice;
7 import com.netease.qa.cloud.nce.utils.CommonData;           import 前应有空行。
8 import com.google.gson.JsonObject;   导入语句'com.google.gson.JsonObject' 字典顺序错误。应在'c
9 import java.io.IOException;          前。
10 import net.sf.json.JSONObject;
11 import org.apache.http.HttpResponse;
12 import org.apache.log4j.Logger;
13 import org.testng.Assert;
14 import org.testng.annotations.*;      不应使用 '*' 形式的导入 - org.testng.an
15
16 public class CreateNamespaceTest {
17     private Logger log = Logger.getLogger(CreateNamespaceTest.class);
18     String urlPath = "api/v1/namespaces/";
19
20
21 @BeforeClass(alwaysRun = true)    缺少 Javadoc。
22     public void setUpClass() throws Exception {
23         log.info("=====setUpClass CreateNamespaceTest=====");
24         Thread.sleep(1000);        'method def' 子元素缩进了5个缩进符，应
25         // 初始化配置文件
26         CommonData.initforOpenAPI_microService();
27         // CommonData.initforWebAPI();
28         log.info("host:" + CommonData.WebOpenAPIHost);
29
30     }
31
32 }
```

```

6 package com.netease.qa.cloud.nce.testcase.openapi.microservice;
7
8 import com.google.gson.JsonObject;
9 import com.netease.qa.cloud.nce.utils.CommonData;
10 import java.io.IOException;
11 import net.sf.json.JSONObject;
12 import org.apache.http.HttpResponse;
13 import org.apache.log4j.Logger;          import org.testng.Assert;
14 import org.testng.annotations.AfterClass;
15 import org.testng.annotations.AfterMethod;
16 import org.testng.annotations.BeforeClass;
17 import org.testng.annotations.BeforeMethod;
18 import org.testng.annotations.Test;
19
20
21 public class CreateNamespaceTest {
22     private Logger log = Logger.getLogger(CreateNamespaceTest.class);
23     String urlPath = "api/v1/namespaces/";
24
25 /**
26 * 测试准备.
27 */
28 @BeforeClass(alwaysRun = true)
29     public void setUpClass() throws Exception {
30         log.info("=====setUpClass CreateNamespaceTest=====");
31         Thread.sleep(1000);
32         // 初始化配置文件
33         CommonData.initforOpenAPI_microService();
34         // CommonData.initforWebAPI();
35         log.info("host:" + CommonData.WebOpenAPIHost);
36     }
37
38 }
```


静态代码检查常用工具-Checkstyle

自定义检查项配置：



静态代码检查常用工具-PMD

特点：PMD 通过其内置的编码规则对 Java 代码进行静态检查，主要包括对潜在的 Bugs，未使用的代码，重复的代码，循环体创建新对象等问题的检验

目标：发现潜在Bugs

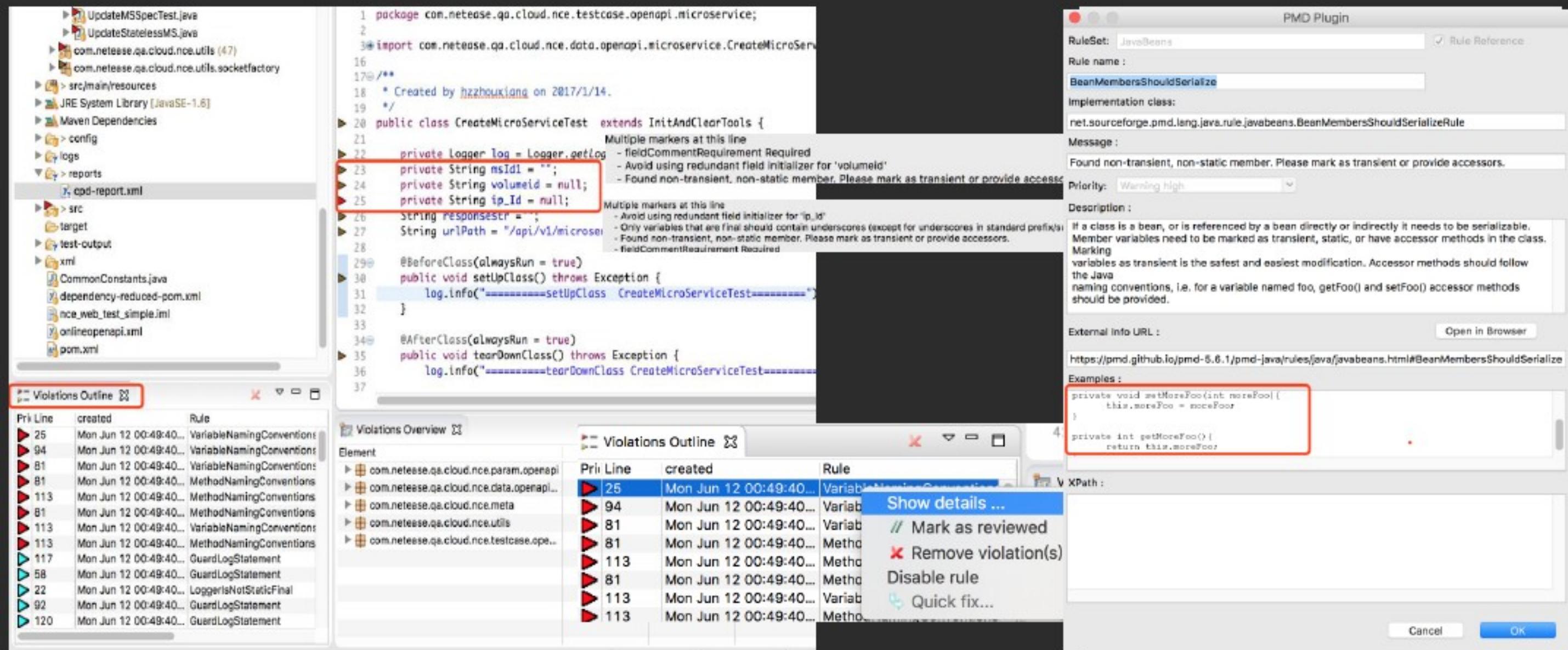
检查项：

- 检查潜在代码错误
- 未使用代码
- 复杂的表达式
- 重复的代码
- 循环体创建新对象
- 重复代码



静态代码检查常用工具-PMD

实例：PMD 将违规事件依重要性依序分为 error high, error, warning high, warning, information 五个等级



The screenshot shows the PMD IDE interface with several windows open:

- Project Explorer:** Shows Java files like `UpdateMSSpecTest.java`, `UpdateStatelessMS.java`, and `CreateMicroServiceTest.java`.
- Code Report:** A file named `cod-report.xml` is selected.
- Violations Outline:** A table showing violations:

Pri	Line	created	Rule
25		Mon Jun 12 00:49:40...	VariableNamingConventions
94		Mon Jun 12 00:49:40...	VariableNamingConventions
81		Mon Jun 12 00:49:40...	VariableNamingConventions
81		Mon Jun 12 00:49:40...	MethodNamingConventions
113		Mon Jun 12 00:49:40...	MethodNamingConventions
81		Mon Jun 12 00:49:40...	MethodNamingConventions
113		Mon Jun 12 00:49:40...	VariableNamingConventions
113		Mon Jun 12 00:49:40...	MethodNamingConventions
117		Mon Jun 12 00:49:40...	GuardLogStatement
58		Mon Jun 12 00:49:40...	GuardLogStatement
22		Mon Jun 12 00:49:40...	LoggerIsNotStaticFinal
92		Mon Jun 12 00:49:40...	GuardLogStatement
120		Mon Jun 12 00:49:40...	GuardLogStatement
- Violations Overview:** A table showing violations by element:

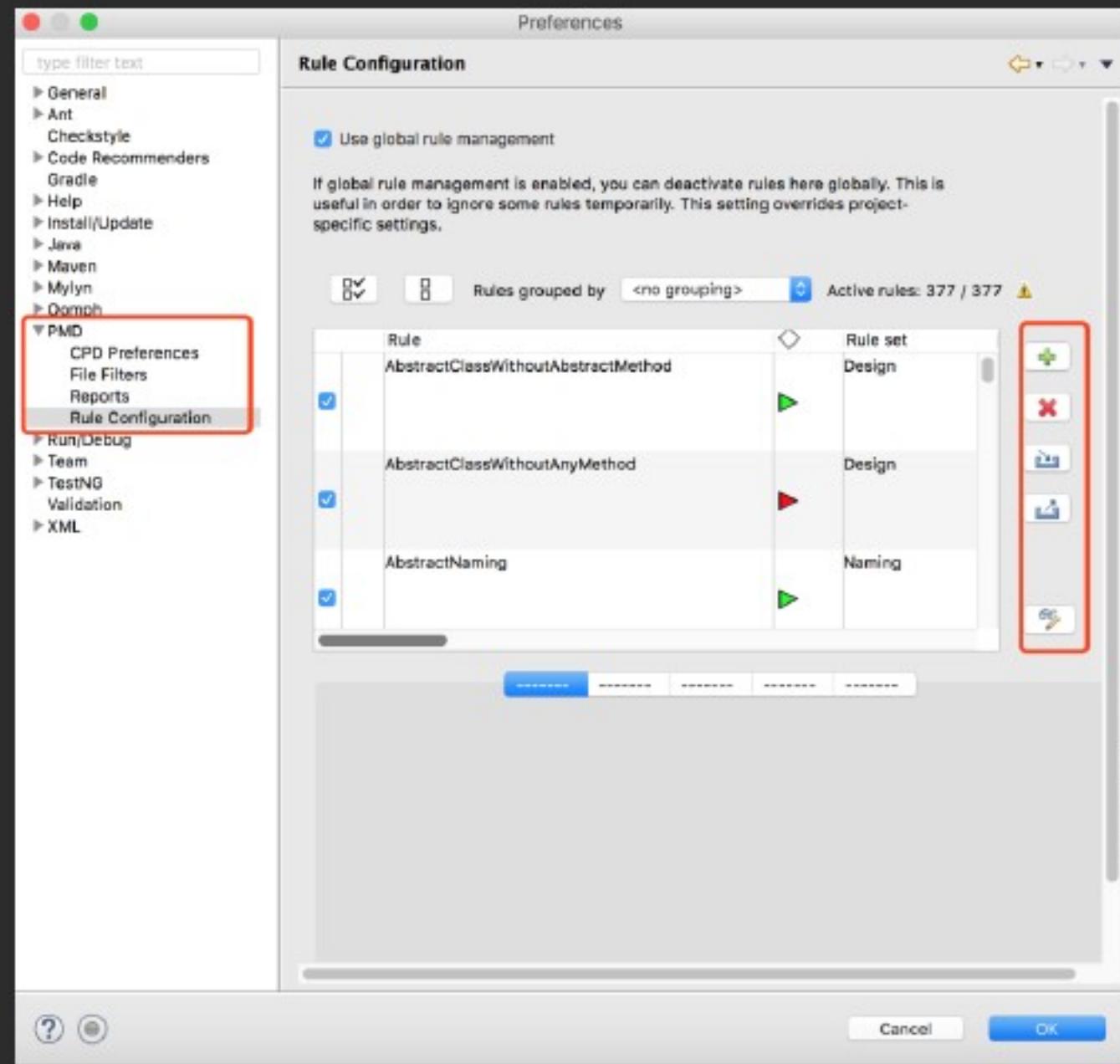
Element	Pri	Line	created	Rule
com.netease.qa.cloud.nce.param.openapi	25		Mon Jun 12 00:49:40...	VariableNamingConventions
com.netease.qa.cloud.nce.data.openapi	94		Mon Jun 12 00:49:40...	VariableNamingConventions
com.netease.qa.cloud.nce.meta	81		Mon Jun 12 00:49:40...	VariableNamingConventions
com.netease.qa.cloud.nce.utils	81		Mon Jun 12 00:49:40...	MethodNamingConventions
com.netease.qa.cloud.nce.testcase.openapi	113		Mon Jun 12 00:49:40...	MethodNamingConventions
- Violations Detail:** A detailed view of a specific violation in `CreateMicroServiceTest.java`. It highlights lines 23-25 and shows the following message:

Multiple markers at this line
 - fieldCommentRequirement Required
 - Avoid using redundant field initializer for 'volumeId'
 - Found non-transient, non-static member. Please mark as transient or provide accessors.
- Rule Configuration:** A dialog window for the rule `BeanMembersShouldSerialize`. It includes fields for RuleSet (JavaBeans), Rule name (BeanMembersShouldSerialize), Implementation class (net.sourceforge.pmd.lang.java.rule.javabeans.BeanMembersShouldSerializeRule), Message (Found non-transient, non-static member. Please mark as transient or provide accessors.), Priority (Warning high), and Description (If a class is a bean, or is referenced by a bean directly or indirectly it needs to be serializable. Member variables need to be marked as transient, static, or have accessor methods in the class. Marking variables as transient is the safest and easiest modification. Accessor methods should follow the Java naming conventions, i.e. for a variable named foo, getFoo() and setFoo() accessor methods should be provided.).
- Code Examples:** Examples for the rule:


```
private void setMoreFoo(int moreFoo) {  
    this.moreFoo = moreFoo  
}  
  
private int getMoreFoo() {  
    return this.moreFoo;  
}
```
- Contextual Menu:** A context menu is open over the violation in the code editor, with options: Show details..., Mark as reviewed, Remove violation(s), Disable rule, and Quick fix... .

静态代码检查常用工具-PMD

自定义检查项配置：



静态代码检查常用工具-FindBugs

特点：与其他静态分析工具（如Checkstyle和PMD）不同，FindBugs不注重样式或者格式，它专注于帮助java工程师提高代码质量以及排除隐含的缺陷

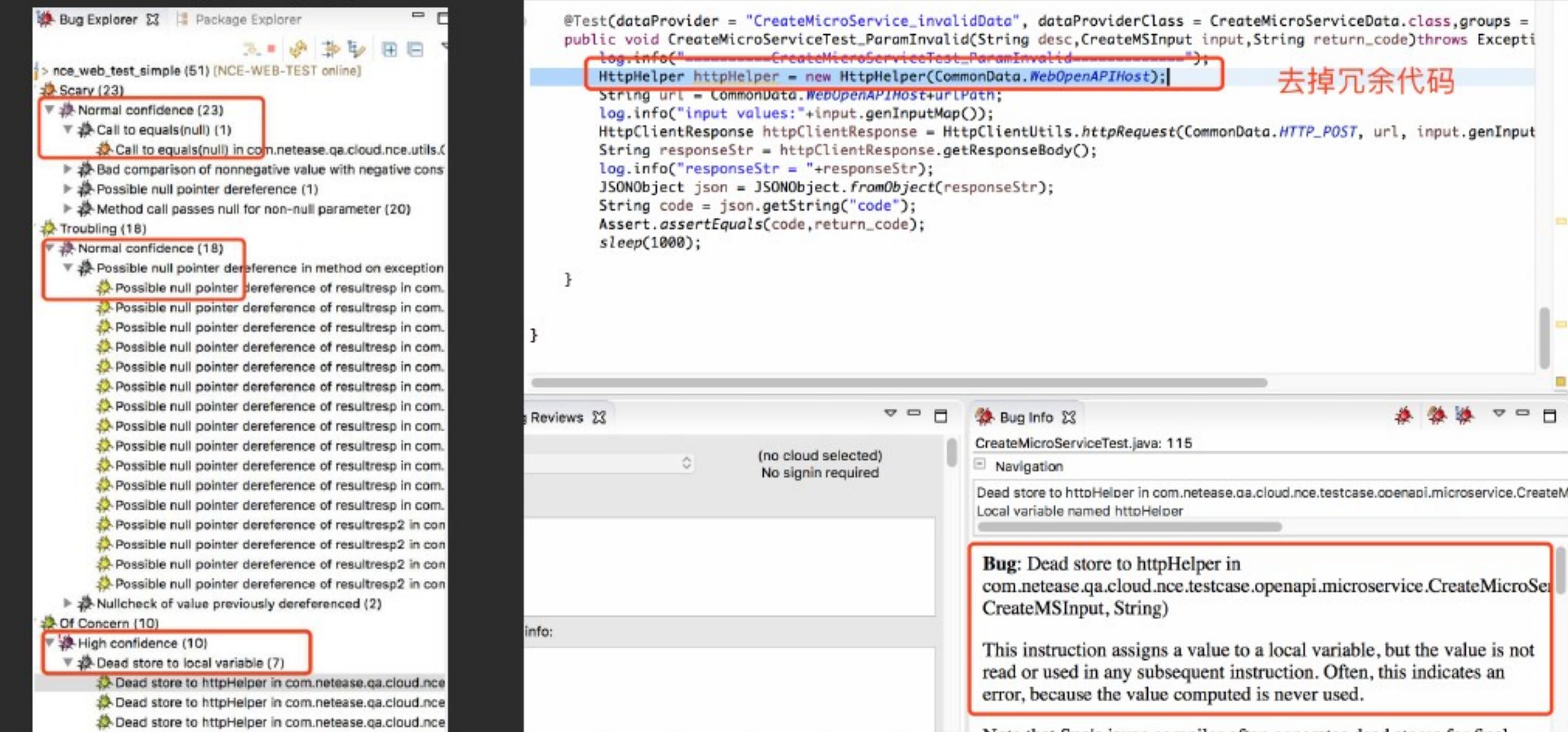
目标：寻找真正的缺陷或者潜在的性能问题

检查项：•正确性•最佳实践反例•多线程正确性 •性能 •安全 •高危



静态代码检查常用工具-FindBugs

实例：按照bug严重程度分为四个等级：Scariest、 Scary、 Troubling和Of Concern四个等级



Bug Explorer

- noe_web_test_simple (51) [NCE-WEB-TEST online]
- Scary (23)
 - Normal confidence (23)
 - Call to equals(null) (1)
 - Call to equals(null) in com.netease.qa.cloud.nce.utils.
 - Bad comparison of nonnegative value with negative cons
 - Possible null pointer dereference (1)
 - Method call passes null for non-null parameter (20)
- Troubling (18)
 - Normal confidence (18)
 - Possible null pointer dereference in method on exception
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp in com.
 - Possible null pointer dereference of resultresp2 in con
 - Possible null pointer dereference of resultresp2 in con
 - Possible null pointer dereference of resultresp2 in con
 - Possible null pointer dereference of resultresp2 in con
 - Nullcheck of value previously dereferenced (2)
- Of Concern (10)
 - High confidence (10)
 - Dead store to local variable (7)
 - Dead store to httpHelper in com.netease.qa.cloud.nce
 - Dead store to httpHelper in com.netease.qa.cloud.nce
 - Dead store to httpHelper in com.netease.qa.cloud.nce

```

    @TestdataProvider = "CreateMicroService_invalidData", dataProviderClass = CreateMicroServiceData.class,groups =
public void CreateMicroServiceTest_ParamInvalid(String desc,CreateMSInput input, String return_code) throws Exception {
    log.info("----- CreateMicroServiceTest_ParamInvalid -----");
    HttpHelper httpHelper = new HttpHelper(CommonData.WebOpenAPIHost);
    String url = CommonData.WebOpenAPIHost+urlPath;
    log.info("input values:" +input.genInputMap());
    HttpClientResponse httpClientResponse = HttpClientUtils.httpRequest(CommonData.HTTP_POST, url, input.genInputMap());
    String responseStr = httpClientResponse.getResponseBody();
    log.info("responseStr = " +responseStr);
    JSONObject json = JSONObject.fromObject(responseStr);
    String code = json.getString("code");
    Assert.assertEquals(code,return_code);
    sleep(1000);
}
}

```

去掉冗余代码

Reviews

(no cloud selected)
No signin required

Bug Info

CreateMicroServiceTest.java: 115

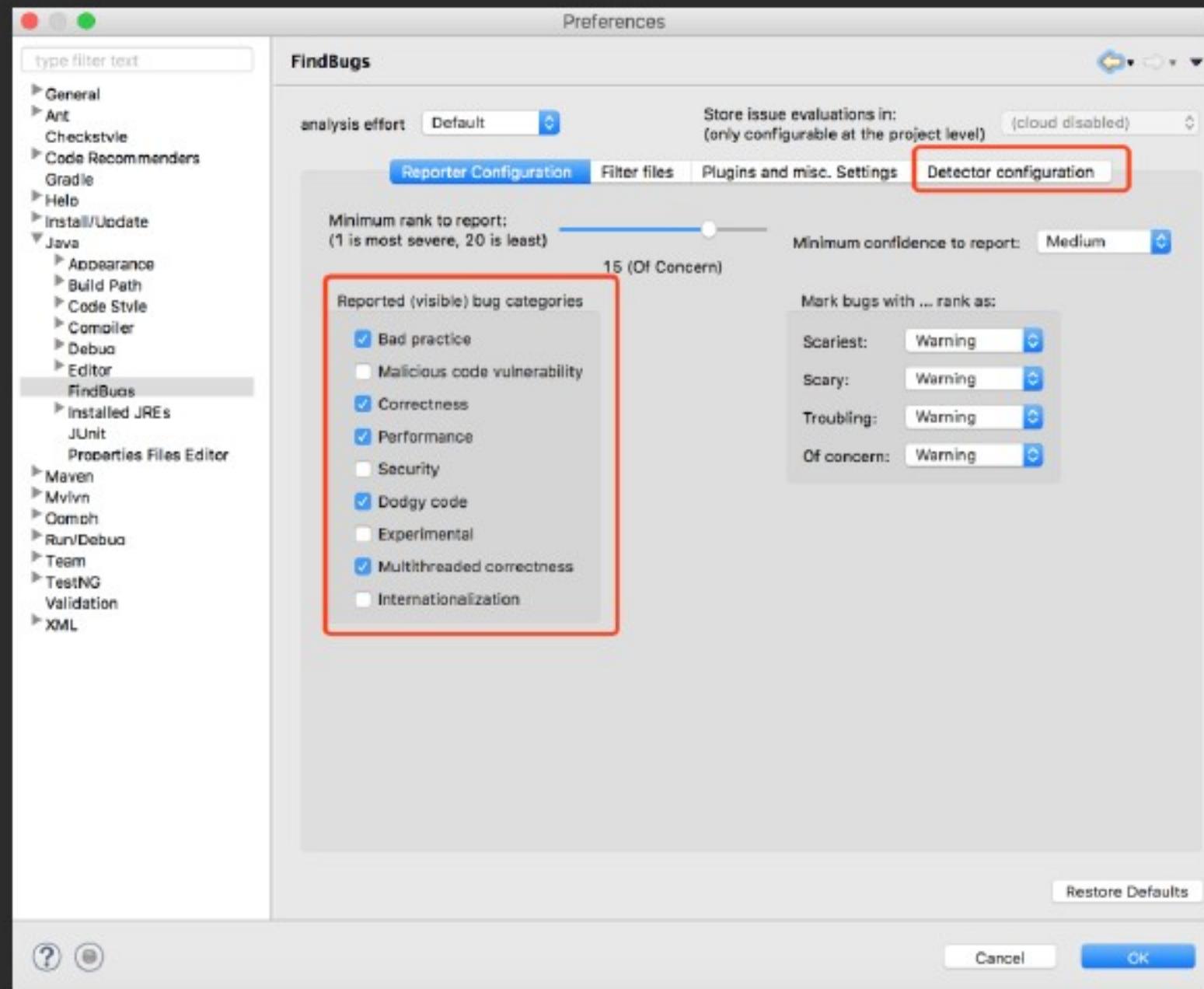
Dead store to httpHelper in com.netease.qa.cloud.nce testcase.openapi.microservice.CreateM...
Local variable named httpHelper

Bug: Dead store to httpHelper in
com.netease.qa.cloud.nce testcase.openapi.microservice.CreateM...
CreateMSInput, String)

This instruction assigns a value to a local variable, but the value is not read or used in any subsequent instruction. Often, this indicates an error, because the value computed is never used.

静态代码检查常用工具-FindBugs

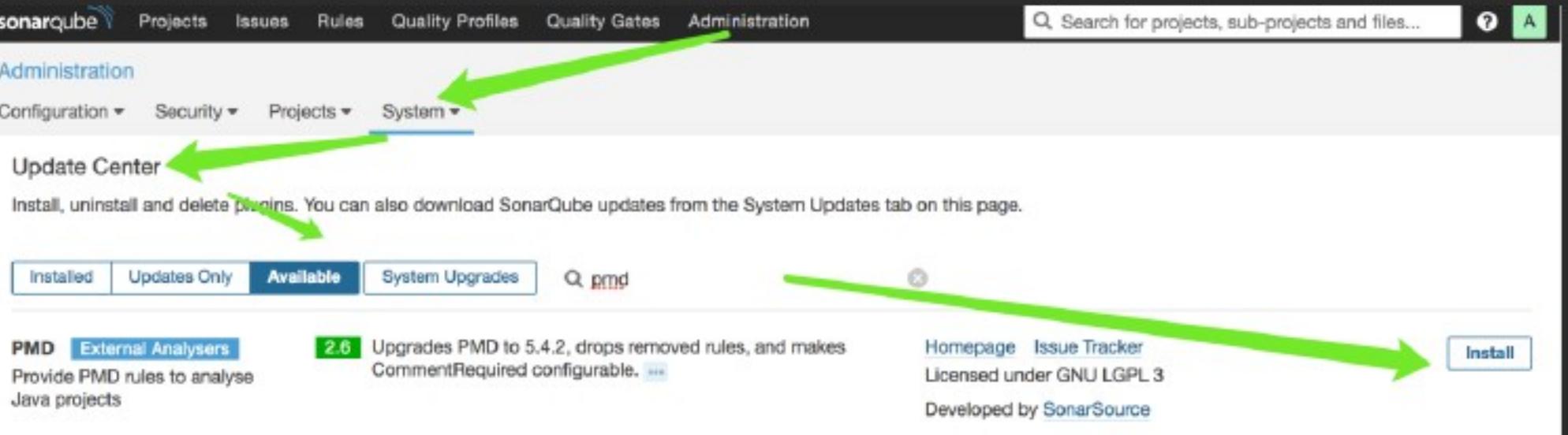
自定义检查项配置：



- JAVA静态代码检查常用工具
 - CheckStyle
 - PMD
 - Findbugs
 - 本地IDE配置及结果分析
- Sonar安装SA 插件及自定义配置
- 静态代码检查原理

Sonar 安装 SA 插件

1. Administration>System>Update Center> Available:



sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects, sub-projects and files... ? A

Administration

Configuration Security Projects System

Update Center

Install, uninstall and delete plugins. You can also download SonarQube updates from the System Updates tab on this page.

Installed Updates Only Available System Upgrades Q pmd

PMD External Analysers 2.6 Upgrades PMD to 5.4.2, drops removed rules, and makes CommentRequired configurable. ... Homepage Issue Tracker Licensed under GNU LGPL 3 Developed by SonarSource

Install

2. 本地下载plugin，重启：

```
sonar
root@sonar-9013:/opt/sonarqube# cd extensions/plugins/
root@sonar-9013:/opt/sonarqube/extensions/plugins# ls -lrht
total 52M
-rw-r--r-- 1 root root 7.1M Jun  2 08:35 sonar-xml-plugin-1.4.2.885.jar
-rw-r--r-- 1 root root 6.3M Jun  2 08:35 sonar-scm-svn-plugin-1.4.0.522.jar
-rw-r--r-- 1 root root 3.1M Jun  2 08:35 sonar-scm-git-plugin-1.2.jar
-rw-r--r-- 1 root root 3.5M Jun  2 08:35 sonar-python-plugin-1.7.0.1195.jar
-rw-r--r-- 1 root root 3.6M Jun  2 08:35 sonar-php-plugin-2.10.0.2087.jar
-rw-r--r-- 1 root root 3.2M Jun  2 08:35 sonar-javascript-plugin-3.0.0.4962.jar
-rw-r--r-- 1 root root 5.9M Jun  2 08:35 sonar-java-plugin-4.9.0.9858.jar
-rw-r--r-- 1 root root 1.6M Jun  2 08:35 sonar-flex-plugin-2.3.jar
-rw-r--r-- 1 root root 11M Jun  2 08:35 sonar-csharp-plugin-5.10.1.1411.jar
-rw-r--r-- 2 root root 129 Jun  2 08:35 README.txt
-rw-r--r-- 1 root root 7.1M Jun 12 05:26 checkstyle-sonar-plugin-3.7.jar
root@sonar-9013:/opt/sonarqube/extensions/plugins#
```

Sonar 安装 SA 插件

安装成功

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects, sub-projects and files... ? A

Administration

Configuration Security Projects System

Update Center

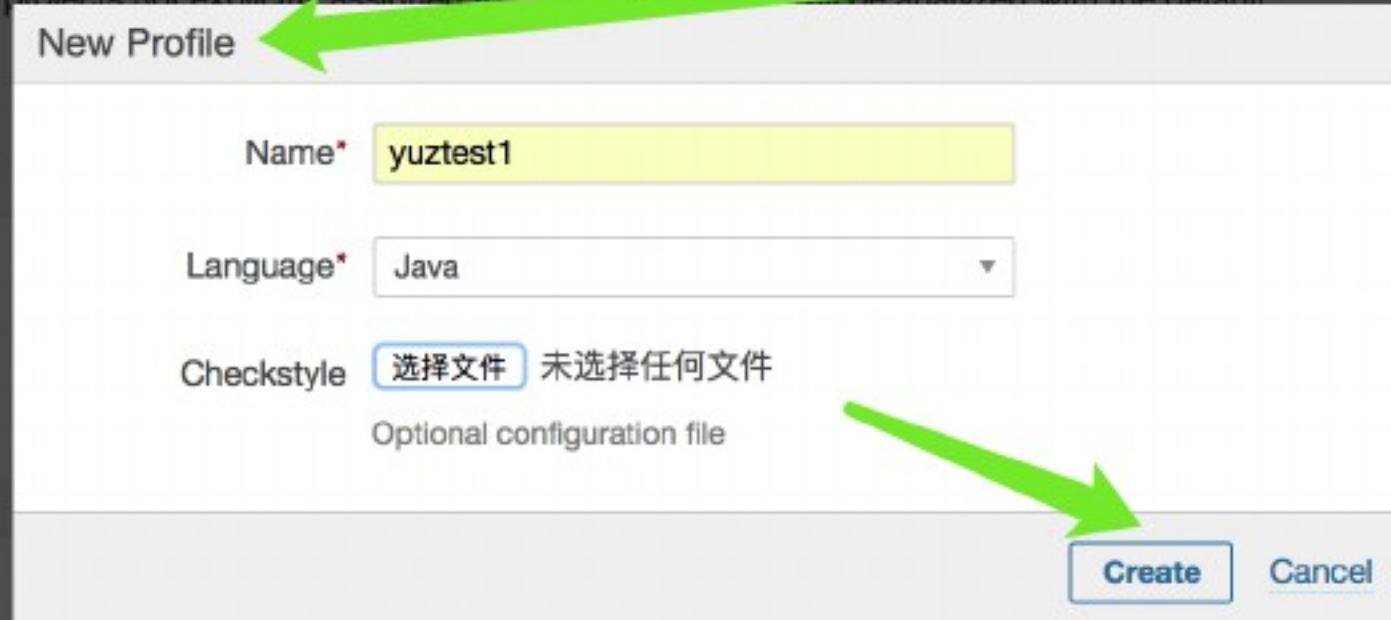
Install, uninstall and delete plugins. You can also download SonarQube updates from the System Updates tab on this page.

Installed Updates Only Available System Upgrades Search

C# Languages Code Analyzer for C#	5.10.1 (build 1411) installed Updates: 5.11 (build 1761) 14 new rules ...	Homepage Issue Tracker Licensed under GNU LGPL 3 Developed by SonarSource	Update to 5.11 (build 1761) Uninstall
Checkstyle External Analyzers Analyze Java code with Checkstyle.	3.7 installed	Issue Tracker Licensed under LGPL-3.0 Developed by Checkstyle	Uninstall
Flex Languages Enables scanning of Flex source files	2.3 installed	Homepage Issue Tracker Licensed under GNU LGPL 3 Developed by SonarSource	Uninstall

Sonar 自定义代码检查配置

创建Quality Profile (可导入配置)



sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects, sub-projects and files... ? A

Quality Profiles

Quality Profiles are collections of rules to apply during an analysis.

For each language there is a default profile. All projects not explicitly assigned to a profile will be analyzed with the default profile.

New Profile

Name* yuztest1

Language* Java

Checkstyle 选择文件 未选择任何文件

Optional configuration file

Create Cancel

All Profiles ▾

C#, 1 profile(s)

Sonar way

Flex, 1 profile(s)

Recently Added Rules

- Modified Control Variable
Java, not yet activated
- Method Length
Java, not yet activated
- Illegal Instantiation
Java, not yet activated

Sonar 自定义代码检查配置

可查看默认配置

sonarcube Projects Issues Rules Quality Profiles Quality Gates Administration Q Search for projects, sub-projects and files... A

Quality Profiles / Java
yuztest1 Updated: Never Used: Never Changelog Actions

Rules	Active	Inactive
Total	0	563
Bugs	0	108
Vulnerabilities	0	33
Code Smells	0	422
Activate More		

Exporters
[Checkstyle](#)

This XML file does not appear to have any style in it.

```
<!-- Generated by Sonar -->
<module name="Checker">
  <module name="SuppressionCommentFilter"/>
  <module name="SuppressWarningsFilter"/>
  <module name="TreeWalker">
    <module name="FileContentsHolder"/>
    <module name="SuppressWarningsHolder"/>
  </module>
</module>
```

Inheritance
yuztest1
yuztest1 Updated: Never Used: Never Changelog Actions

sonarcube 仪表盘 问题 指标 Rules 质量配置 Quality Gates More

质量配置
Show: All Profiles

BACKEND_CODE_GUIDE_V1.0 Java

代码规则
86 active rules

- 0 阻断
- 2 次要
- 28 严重
- 9 提示
- 47 主要

永久链接
 FindBugs
 Android Lint
 Checkstyle
 PMD

Projects
No project

C Sonar way

C++ C++_CODE_GUIDE_V1.0

Sonar way 0 projects

Java

Android Lint 0 projects

ATM_RULES 1 projects

[BACKEND_CODE_GUIDE_V1.0](#)

BACKEND_CODE_GUIDE_V1.0 beauty-V0.1 1 projects

C 0 projects

cloud-java-code-guide-v1.0 0 projects

cloud-java-backend-new 0 projects

Edu 20 projects

EduAndroid 3 projects

EduBackendJava 0 projects

FindBugs 0 projects

FindBugs Security Audit 0 projects

FindBugs Security Minimal 0 projects

配置继承

Edu 86 active rules

BACKEND_CODE_GUIDE_V1.0 86 active rules

nss-v0.1 86 active rules 2 overridden rules

修改日志
修改记录 可选 至 可选 搜索

比较
With Android Lint 比较

Sonar 自定义代码检查配置

自定义规则和优先级

sonarcube Projects Issues Rules Quality Profiles Qual sonarcube Projects Issues Rules Quality Profiles Quality Gates Administration Q Search for projects, sub-projects and files... ? A

For each language there is a default profile. All projects not explicitly a

All Profiles ▾

C#, 1 profile(s)	Projects	Language	Rules
Sonar way	Default	<input checked="" type="checkbox"/> Java	563
		Python	238
		C#	229
		JavaScript	186
		PHP	127
		Flex	79
		XML	13

Flex, 1 profile(s)	Projects	Language	Rules
Sonar way	Default		

Java, 2 profile(s)	Projects	Rules	Updated	Used
Sonar way	Default	271	Never	Never
yuztest1	0	0	Never	Never

JavaScript, 2 profile(s)	Projects	Rules	Updated	Used
115.238.125.154:9000/coding_rules#qprofile=AvymylJFjgyIRVz...				

Activate in Quality Profile

Quality Profile: yuztest1

Severity: Major

Activate Cancel

"@Override" should be used on overriding and implementing methods

"@RequestMapping" methods should be "public"

Activate More Rules Back up Compare

A large green arrow points from the "Activate" button in the modal dialog to the "Activate" button in the main interface.

- JAVA静态代码检查常用工具
 - CheckStyle
 - PMD
 - Findbugs
 - 本地IDE配置及结果分析
- Sonar安装SA 插件及自定义配置
- 静态代码检查原理

静态代码检查原理

- 1) 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。
- 2) 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。

Java 静态分析工具	分析对象	应用技术
Checkstyle	Java 源文件	缺陷模式匹配
FindBugs	字节码	缺陷模式匹配；数据流分析
PMD	Java 源代码	缺陷模式匹配

THANK YOU !
