For Java8 environment, share an example to create PMD ruleset to find the call hierarchy as stand alone java project , step by step guide on Java and PMD cli interfacce to invoke call hierarchy rule on Source Code located in other Physical Folder of the machien where CalHeirarchy rule there

Sure! Here's a step-by-step guide to create a PMD ruleset to find the call hierarchy in a Java project using Java 8 and the PMD CLI interface:

**Step 1: Create a Custom Rule**

1. **Write a Java class** that implements the Rule interface. You can extend the AbstractJavaRule class to ease the process.
2. **Override the visit method** for the specific node type you want to check. For example, if you want to check method calls, you can override the visit(ASTMethodCall node, Object data) method.

Here's a simple example:

java

import net.sourceforge.pmd.AbstractJavaRule;

import net.sourceforge.pmd.lang.java.ast.ASTMethodCall;

import net.sourceforge.pmd.RuleContext;

import net.sourceforge.pmd.Report;

public class CallHierarchyRule extends AbstractJavaRule {

@Override

public Object visit(ASTMethodCall node, Object data) {

Report report = (Report) data;

report.addViolation(this, node);

return super.visit(node, data);

}

}

**Step 2: Compile the Custom Rule**

1. **Compile the Java class** and create a JAR file. You can use Maven or any other build tool for this.

**Step 3: Create a Ruleset XML File**

1. **Create a new XML file** for your ruleset. Here's a template:

xml

<?xml version="1.0" encoding="UTF-8"?>

<ruleset name="Custom Rules" xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0 https://pmd.sourceforge.io/ruleset\_2\_0\_0.xsd">

<description>My custom rules</description>

<rule ref="path/to/your/custom/rule.jar/CallHierarchyRule"/>

</ruleset>

**Step 4: Run PMD CLI with the Custom Ruleset**

1. **Navigate to the directory** where your source code is located.
2. **Run the PMD CLI command** to execute the ruleset on your source code. For example:

sh

pmd -d /path/to/your/source/code -f text -R /path/to/your/ruleset.xml

This command will analyze the source code in the specified directory using the custom ruleset and output the results in text format.

**Step 5: Analyze the Results**

1. **Review the output** to see the violations reported by your custom rule.

That's it! You've created a custom PMD ruleset to find the call hierarchy in your Java project. If you have any questions or need further assistance, feel free to ask!

1pmd.github.io2pmd.github.io3pmd.github.io4github.com

Extend the same to pass methodName for which teh CallHierarchy to be displayed

Got it! Let's extend the example to pass a method name for which the call hierarchy should be displayed. We can do this by adding a parameter to the custom rule that specifies the method name, then modifying the rule to check for this method name. Here’s the updated step-by-step guide:

**Step 1: Create a Custom Rule with Method Name Parameter**

1. **Write a Java class** that extends the AbstractJavaRule class. Add a parameter for the method name you want to check.

Here's an example:

java

import net.sourceforge.pmd.AbstractJavaRule;

import net.sourceforge.pmd.lang.java.ast.ASTMethodCall;

import net.sourceforge.pmd.RuleContext;

import net.sourceforge.pmd.Report;

import net.sourceforge.pmd.properties.StringProperty;

public class CallHierarchyRule extends AbstractJavaRule {

private static final StringProperty METHOD\_NAME = new StringProperty("methodName", "Method name to check", "", 1.0f);

public CallHierarchyRule() {

definePropertyDescriptor(METHOD\_NAME);

}

@Override

public Object visit(ASTMethodCall node, Object data) {

String methodName = getProperty(METHOD\_NAME);

if (node.getMethodName().equals(methodName)) {

Report report = (Report) data;

report.addViolation(this, node);

}

return super.visit(node, data);

}

}

**Step 2: Compile the Custom Rule**

1. **Compile the Java class** and create a JAR file. You can use Maven or any other build tool for this.

**Step 3: Create a Ruleset XML File with Method Name Parameter**

1. **Create a new XML file** for your ruleset and specify the method name as a property.

Here's a template:

xml

<?xml version="1.0" encoding="UTF-8"?>

<ruleset name="Custom Rules" xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0 https://pmd.sourceforge.io/ruleset\_2\_0\_0.xsd">

<description>My custom rules</description>

<rule ref="path/to/your/custom/rule.jar/CallHierarchyRule">

<properties>

<property name="methodName" value="myMethodName"/>

</properties>

</rule>

</ruleset>

**Step 4: Run PMD CLI with the Custom Ruleset**

1. **Navigate to the directory** where your source code is located.
2. **Run the PMD CLI command** to execute the ruleset on your source code, passing the method name as a property in the ruleset XML.

For example:

sh

pmd -d /path/to/your/source/code -f text -R /path/to/your/ruleset.xml

This command will analyze the source code in the specified directory using the custom ruleset and output the results in text format.

**Step 5: Analyze the Results**

1. **Review the output** to see the violations reported by your custom rule for the specified method name.

That's it! You've extended the custom PMD ruleset to include a method name parameter for displaying the call hierarchy. If you have any questions or need further assistance, feel free to ask

Message Copilot