Utilizing the Inspection Tool

Summary

This guide will describe the eGovFrame's Code Inspection tool called PMD and its basic usage.

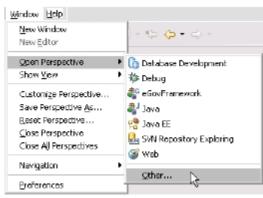
Basic manual

You can run Code Inspection from the IDE's PMD Perspective in order to batch-inspect code conveniently.

Switching to PMD Perpective

Following is how to switch to the PMD Perspective.

1. Choose 'Window' > 'Open Perspective' > 'Other...' from the menu



2. Choose PMD from the Open Perspective window

🗧 Open Perspective	_ 🗆 🔀
Aspect Visualization CVS Repository Exploring Database Debug COS Repository Exploring Database Development COS Repositor Cos Framework Cos Framework	
FMD Benote System Explorer Besource SVN Repository Exploring	Carcel

3. After switching to the PMD Perspective, you will see Package Explorer, Violation Outline, Violation Overview and other views within the IDE.

🗧 PND - Eclipse Flatform	_		_	_	_	_	_ 🗆 🔀
Die Edit Source Refactor Navigate Gearch	Brotect Run Window	Eep					
🔁 • 🖾 🖄 • 🐼 • 🖓 • 🖓 •	4.0	$+ \langle g \rangle + \langle \phi \rangle + \langle \phi \rangle + \langle g \rangle + \langle g \rangle + \langle \phi \rangle + \langle $				🖻 📔 🔒	a 🌮 😢
🛱 Package Explorer 🖄 🛄 🖂							
Eenversi Serversi Serversi							
L Violations Burline 23							
	Tiglations Overview 3	23					
	Element		# Violations	# Violations/LDC	♥ #Violatio	Project	1
					1		

Running code inspection

Choose a project to inspect.

In the Package Explorer, right-click on the project, and choose PMD > Check Code with PMD in the context menu.

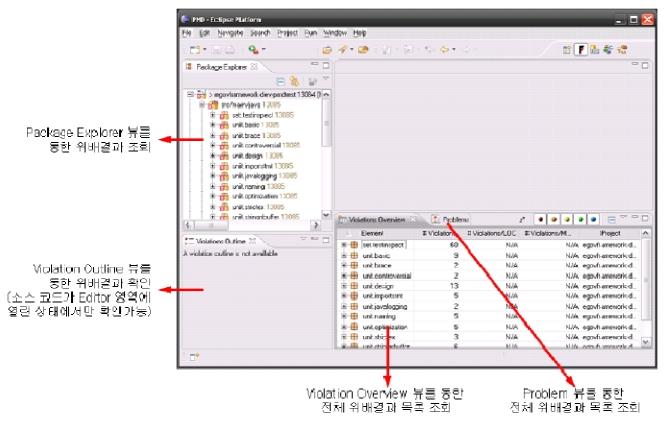
🖶 PMD - Eclipse Platform	🚵 Import			
Eile Edit Source Refactor Novigab				
📫 • 🔜 🗁 🕯 🎋 • 💽 • 🤮	open Project Close Project	F5	1P	
문 😫 Segovframework-dav-prodes 1 환 🧗 so/main/java 13385	Close Unrelated Pr			
- 🔄 src/main/resources 13085	Bun As			
🚰 grc/text/java 13085 (7年 grc/text/resources 13085	<u>D</u> ebug As Profile As			
I A State State Library [dk1.5.0				
🛱 🕌 Maven Dependencies	Validate			
E - 2⊋ reports E - 2⊕ src 13085	m2 Mayen			
Biranget 13108	Team		- 1	
- 🛐 pom.xml 13107	Compare With Replace With			
M Ruleset_egov_eng.xml 130	Restore from Loca	History		
- M Ruleset_egov_kor.xml 1311 - M Ruleset_egov.xml	Spring Tools			
B B Servers	Web Development	: Tools		
🖻 📷 > swalegovirame-template3 24 [- 1	
· 👕 swa.egovframe-template9	Aspect <u>1</u> Tools PMD			Generate reports
<	Properties	Alt+Enter	P	Clear violations reviews Find Suspect Cut And Paste
🗮 Violations Outline 🔀			į	Clear PMD Molations
A violation outline is not available	🛅 Violati	ons Overview 😒	P	Check Code With PMD
	Element			# Violations # Violations

• The entire project's source code will be inspected. If you want to run inspections only on individual source files, apply the same step as above on the files from the Package Explorer.

Only Java source codes will run through the inspection, and the following will be excluded.

Inspection results

After the inspection, you can check the results and see the violating codes. PMD Perspective provides multiple views for you to see the inspection results.



Package Explorer

If there are violating code areas within the project, similar to compile errors, the project icon and the violating source icons in the Package Explorer will display red X boxes as below. (📅 🕮)

These icons will persist until the violations have been corrected. Problem view

Problem view will list the violating lines in the source code. Open the Problem view in the PMD Perspective as follows.

1. Window > Show View > Other... in the Eclipse menu

Window Help		
<u>New Window</u> New <u>E</u> ditor		1
open Perspective	Þ	
Show <u>M</u> ew	Þ	gther At+shift+Q, Q
Customize Perspective		k k
Save Perspective As		
Reset Perspective		
⊆lose Perspective		
Close All Perspectives		
Navigation	۲	
Breferiences		

2. In the Show View dialog, choose Problem under the General category.

🖶 Show View 📃 🗖 🔀
type filter text
General General General Gassic Search Gastarch Gastarch Gassic Search
Use F2 to display the description for a selected view.
OK Carcel

Use the Problem view to see the violating lines, and access them in the editor to fix the violated source codes.

Category	Description				
Description	Detailed description of the violation				
Resource	Violating file name				
Path	Violating file's path				
Location	Violating line number				

Problems 23					2
5 errors, 12 warnings, 0 others Description	Resource	Path 👻	Location	Туре	
🖻 🔞 Errars (45 items)					
🛞 'howdy' 처럼 사용되지 않는 'methad' 파라미터가 있음	UnusedFormalParameter, java	egowuletest/src/	line 5	PMD Marker	
📀 사용되지 않는 Private field 11 가 선언되었음	UnusedPrivateField, java	egowruletest/src/	line 5	PMD Marker	
👴 사용되지 않는 Private Method (foo()) 가 선언되었음	UnusedPrivateMethod.java	agovruletest/src/	line 5	PMD Marker	
🔞 생성자에서 Assign된 변수 11를 Final로 선언하지 않았음	UnusedPrivateField.java	agovruletest/src/	line 5	PMD Marker	
📀 🔞 Empty String 을 체코하기 위해 String.trim(), length() 을 사용하는	InefficientEmptyStringCheck, java	egowruletest/src/	line 6	PMD Marker	
O String 객체에서 toString()함수를 사용하는 것은 불필요함	StringToString, java	egowruletest/src/	line 8	PMD Marker	
🤢 String 볼 append 할 경우, String,valueOf 함수를 사용할 필요 없;	UselessStringValueOt, java	egovruletest/src/	line 7	PMD Marker	
🤨 StringBuffer 할수에서 nonliteral 을 직접 concatenate 하지 말 것	InefficientStringButtering.java	agovruletest/src/	line 7	PMD Marker	
◎ 필요없는 Instance가 생성되어 있음	StringInstantiation, java	egowuletest/src/	line 6	PMD Marker	

Double-clicking an entry will jump to the violating line in the Editor.

Violations Overview view

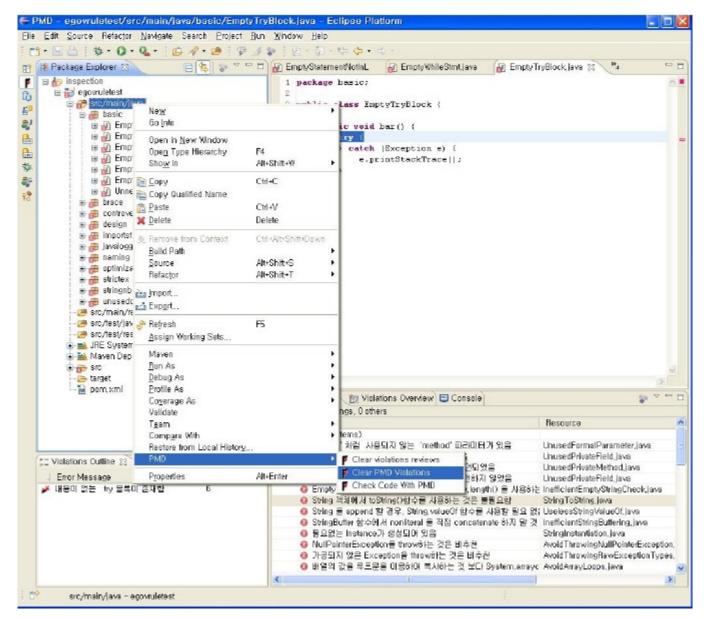
Using Violations Overview will be described in the Reporting inspection results section later on.

Resetting inspection results

Inspection results will persist on the source until you edit them. You may need to reset the inspection results because it may be confusing to distinguish between inspection results and code compile error results. Or, if you want to re-run inspection, you need to reset inspection results.

In order to reset inspection results, choose a project and reset it as below.

In the Package Explorer, right-click on the project, and choose PMD > Clear PMD Violations from the context menu.



This will reset the inspection results, and will hide the Problem view, Violations Overview view, and Package Explorer view in the IDE.

eGovFrame standard inspection rules

eGovFrame defines total of 39 rules for Code Inspection in terms of logical/phrase/reference errors. eGovFrame rule set needs to be installed according to the standard installation guideline; each rule will be explained in the following sections, as well as examples of violating codes, remedies, and compliant codes.

Rule#01. EmptyCatchBlock

public void doSomething() {	
try	
{	
FileInputStream fis = new FileInputStream("/tmp/bugger");	
}	
catch (IOException ioe)	
{	
//	
}	
}	

Rule#02. EmptyIfStmt

Description: empty if conditions

Recommendation: avoid empty if conditions

Rule#03. EmptyWhileStmt

- Description: empty while conditions
- Problematic code:

Recommendation: avoid empty while conditions

Rule#04. EmptyTryBlock

Description: empty try conditions

Problematic code:

public class Foo {	1
public void bar() {	1
try	ł
	ł
}	ł
catch (Exception e)	ł
	ł
e.printStackTrace();	į
}	ł
}	į
}	ł
	j
Recommendation: avoid empty try conditions	

Rule#05. EmptyFinallyBlock

Description: empty finally statements

Problematic code:

p	public class Foo {				 	 	
	public void bar() {						
	try						
	{						
	int x=3;						
	}						
	finally						
	{						
	//						
	}						
	}						
}	ł						
1.					 	 	
• D	acommondation	there must not be	any omnty final	ly statmonts			

Recommendation: there must not be any empty finally statments

ule#06. UnnecessaryConversionTemporary

:	Unnecessary String conversions into temporary variables
	Problematic code:



Rule#07. EmptyStatementNotInLoop

- Description: unnecessary semi-colons
- Problematic code:

```
public class MyClass {
    public void doit() {
        :
        System.out.println("look at the extra semicolon");;
    }
}
```

Recommendation: avoid using empty statements with semi-colons only.

Rule#08. WhileLoopsMustUseBraces

- Description: bracket-less while statements
- Problematic code:

public void doSomething() {
while (true)
x++;
}

Recommendation: avoid using while statements without brackets

Rule#09. AssignmentInOperand

- Description: assignments within conditions/operands
- Problematic code:

public class Foo {	1
public void bar() {	
int x = 2;	
if $((x = getX()) = 3)$ {	
System.out.println("3!");	
}	
}	
private int getX() {	
return 3;	
)	
Recommendation: avoid using assignments in conditions/operands, because it decreases code readability and increases complixity	

Rule#10. UnnecessaryParentheses

- Description: unnecessary parentheses
- Problematic code:

public class Foo {
 boolean bar() {
 return (true);
 }
}
Recommendation: avoid unnecessary parentheses, because it decreases code readability

Rule#11. SimplifyBooleanExpressions

- Description: unnecessary boolean comparisons
- Problematic code:

```
private boolean bar = (isFoo() == true);
public isFoo() {
    return false;
    }
}
• Recommendation: avoid using unnecessary boolean comparisons
```

Rule#12. SwitchStmtsShouldHaveDefault

- Description: default-less switch statements
- Problematic code:

```
public class Foo {
    public void bar() {
        int x = 2;
        switch (x) {
            case 2;
            int j = 8;
        }
    }
}
```

Recommendation: always define switch label in switch statements

Rule#13. AvoidReassigningParameters

- Description: reassigning parameters
- Problematic code:

Du	blic class Foo {	
Ĺ	private void foo(String bar) {	
	bar = "something else";	
	}	
}		
• Re	commendation: parameters shall not be modified in value.	

Rule#14. FinalFieldCouldBeStatic

- Description: using static instead of final
- Problematic code:

public class Foo {
 public final int BAR = 42;
}
Recommendation: switching to static instead of final can reduce overhead

Rule#15. EqualsNull

Description: null comparison using equals()

```
Problematic code:
```

class Bar {
 void foo() {
 String x = "foo";
 if (x.equals(null)) {
 doSomething();
 }
 }
}
Recommendation: equals method shall not be used for comparing null values

Rule#16. SimpleDateFormatNeedsLocale

- Description: locale-less SimpleDateFormat
- Problematic code:

......

public class Foo {
 private SimpleDateFormat sdf = new SimpleDateFormat("pattern");
}

Recommendation: always assign locale when using SimpleDateFormat

Rule#17. ImmutableField

Description: use final for constructor variables

Rule#18. AssignmentToNonFinalStatic

- Description: erroneous usage of static type
- Problematic code:

```
public class StaticField {
    static int x;
    public FinalFields(int y) {
        x = y;
    }
}
Recommendation: avoid using static fields in unsafe manner
```

Rule#19. AvoidSynchronizedAtMethodLevel

- Description: overusing synchronization at method level
- Problematic code:

```
public class Foo {
    synchronized void foo() {
    }
}
Recommendation: use synchronization only for blocks rather than methods
```

Rule#20. AbstractClassWithoutAbstractMethod

Description: defining abstract class without abstract methods

Problematic code:

```
public abstract class Foo {
    void int method1() {
        // ...
    }
    void int method2() {
        // ...
    }
}
Recommendation: always define abstract methods in abstract classes
```

Rule#21. UncommentedEmptyMethod

Description: comment-less empty methods

Problematic code:

; } Recommendation: always indicate empty methods with comments

Rule#22. AvoidConstantsInterface

- Description: using constants in interfaces
- Problematic code:

'	Tobendate code.
	public interface ConstantsInterface {
	public static final int CONSTANT1 = 0;
	public static final String CONSTANT2 = "1";
	1

Recommendation: use interfaces only for defining class behaviors

Rule#23. DuplicateImports

Description: duplicate import statements

•	Problematic code:
	import java.lang.String; import java.lang.*;
	import java.lang.*;
	public class Foo {
	·

Recommendation: avoid using duplicate import statements

Rule#24. ImportFromSamePackage

Description: importing from same package

•	Problematic code:
	package foo;
	package foo; import foo.Buz; import foo.*;
	import foo.*;
	public class Bar {
]
	1
	Description of the second description of the second s

Recommendation: avoid importing from same package

Rule#25. SystemPrintln

.

Description: using System.out.print

•	Problematic code:
	class Foo{
	public void testA () {
	System.out.println("Entering test");
	}
	}
•	Recommendation: avoid System.out.print, instead use customized message printing

Rule#26. VariableNamingConventions

	Description: under-bars in variable names
•	Problematic code:
	public class Foo {
	public static final int MY_NUM = 0;
	public String myTest = "";
	DataModule dmTest = new DataModule();
	}
	Recommendation: do not include underlines for variables that are not final

_

Description: using erroneous prefixes for variables

```
Problematic code:
```

```
public class Foo {
    public void bar(String m_baz) {
        int m_boz = 42;
    }
}
Recommendation: avoid using m_ prefixes for non-fields
```

Rule#28. AvoidArrayLoops

- Description: loops for copying arrays
- Problematic code:

```
public class Test {
    public void bar() {
        int[] a = new int[10];
        int[] b = new int[10];
        for (int i=0;i<10;i++) {
            b[i]=a[i];
        }
    }
}</pre>
```

Recommendation: avoid loops and instead use System.arraycopy() for array copies

Rule#29. UnnecessaryWrapperObjectCreation

Description: unnecessary WrapperObjects

•	Problematic code:
	public int convert(String s) {
	int i, i2;
	i = Integer.valueOf(s).intValue();
	i2 = Integer.valueOf(i).intValue();
	<u>.</u>

Recommendation: use custom parse-related methods instead

Rule#30. AvoidThrowingRawExceptionTypes

raw exception types

•	Problematic code:
	public class Foo {
	public void bar() throws Exception {
	throw new Exception();
	}
	1
	Recommendation: use more specific exceptions

Rule#31. AvoidThrowingNullPointerException

- Description: using null pointer exceptions
- Problematic code:
 public class Foo {
 void bar() {
 throw new NullPointerException();
 }
 }
 Recommendation: avoid using NullPointerException

Rule#32. StringInstantiation

- Description: using unnecessary String instances
- Problematic code:

public class Foo {	
private String bar = new String("bar");	
}	
1	

Recommendation: use more simple variables

Rule#33. StringToString

- Description: using toString() on a String instance
- Problematic code:

```
public class Foo {
    private String baz() {
        String bar = "howdy";
        return bar.toString();
    }
}
```

Recommendation: avoid calling toString() on String instances

Rule#34. InefficientStringBuffering

- Description: combining strings inside StringBuffer() constructor
- Problematic code:

```
StringBuffer sb = new StringBuffer( "tmp =" +
System.getProperty("java.io.tmpdir") );
```

Recommendation: use append method instead

Rule#35. InefficientEmptyStringCheck

- Description: using null check or zero size check on Strings
- Problematic code:

public class Foo {
void bar(String string) {
if (string != null && string.trim().size() > 0) {
doSomething();
}
}
L
Recommendation: use custom logic instead to differentiate whitespace and non-whitespace

Rule#36. UselessStringValueOf

- Description: using String.valueOf() when appending
- Problematic code:

public String convert(int i) {
String s;
s = "a" + String.valueOf(i);
return s;
}
·
Recommendation: avoid calling String.valueOf() when appending

Rule#37. UnusedPrivateField

Description: unused private fields

Problematic code:

public class Something {	
private static int FOO = 2; // Unused	
private int i = 5; // Unused	
private int j = 6;	
public int addOne() {	
return j++;	
]	

Recommendation: avoid unused private fields

Rule#38. UnusedPrivateMethod

- Description: unused private methods
- Problematic code:

 public class Something {

 private void foo() {} // unused
 - Recommendation: avoid unused private methods

Rule#39. UnusedFormalParameter

- Description: unused method parameters
- Problematic code:

public class Foo {	
private void bar(String howdy) {	
// howdy is not used	
}	
]	

Recommendation: avoid unused method parameters

Reporting inspection results

You can aggregate and report inspection results as below.

- Personal IDE: Check results in Violations Overview of Eclipse IDE or use file-based reporting in CSV¹), HTML, TXT, or XML.
- Server IDE: Using CI server, Hudson^{2),} Hudson PMD Hudson's PMD Plugin can be used for inspection reports. See eGovFrame's Hudson Code Inspection Tools guide.

Reports in Violations Overview

Inspection results can be checked immediately in the IDE as below.

/ Element	# Violations	# Violations/LOC	# Violations/Method	Project	1
🖶 sattestinspact	44	235.6 / 1000	1.10	egovframework-dev-pridtest	
🖶 unit basic	8	181.8 / 1000	1.14	egovframework-dev-pmdtest	
🚊 🔟 EmptyCatchBlock.java	1	142.9 / 1000	1.00	egovframework-dev-pmdtest	
EmptyCatchBlock	1	142.9 / 1000	1.00	egovframework-dev-pmdtest	
🕸 🕖 EmptyFinaly8 lock.java	1	142.9 / 1000	1.00	egovframework-dev-pmdtest	6
🖻 - 🗾 EmptylfStmt.java	1	200.0 / 1000	1.00	egov/framework-dev-pmdtest	
🖨 🕖 EmptyStatementNotInLoop.java	2	285.7 / 1000	2.00	egovframework-dev-pmdtest	
Finally Statement NotinLoop	2	285.7 / 1000	2.00	egovframework-dev-pmdtest	
🕸 🕖 EmptyTryBlock.java	1	142.9 / 1000	1.00	egovframework-dev-pmdtest	
🟚 🚺 EmptyWhileStrit.java	1	200.0 / 1000	1.00	egovframework-dev-pridtest	5
E. D Henoson auConuning Tomparat	- 1	169 7 / 1000	1.00	onoutranounds dou podtost	

Violations Overview displays a grid for listing statistic violating codes, and control buttons to control the grid on the top right.

Button functionalities

- Z^{*} Statistics button: refreshes the statistics grid. Should be used after using the priority button, or any violation is fixed.
 - Priority buttons: filters violating items in the grid. Five, different-colored buttons that denote from 1 to 5 (low to high, from left to right). 1s and 2s (red and orange) are must-fixes. Each button are toggle-style buttons.
 - Collapse statistics button: collapses all the statistics items in the grid to initial state.
 - View menu: selects sorting method for the grid. The options are as follows:
 - Show violations to packages: sorts violating items by packages.
 - Show violations to files: sorts violating items by files.
 - Show packages with files: sort by packages, along with displaying file information.

- Element: violating packages, files, and detailed information.
- Violations: violation count for the selected element.
- ³⁾
 Violations/LOC: parts-per-thousand (per-mil), as in violations per 1,000 lines of code.
 Violations/Method: violations per method, or average violations per method for package elements.
 Project: name of project that the selected violating element belongs to.

These statistical information can be utilized by the developer, directly from the IDE, for improving code quality.

Per-file report generation

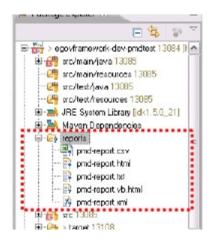
To generate per-file reports, choose a project, then proceed as follows.

• In Package Explorer, choose Project, then right-click on it.

		pty FryBlock, Java -	Eclipse Platform			
- III - 1	tor <u>N</u> avigate Search <u>Project</u>	t <u>B</u> un <u>W</u> indow <u>H</u> e	þ			
) • Q. • 1 🕒 🛷 • 😕 🕴	日月 新日日 - 日) - 🎨 😓 -			
😫 Package Explorer	3 🛛 🔁 🕼 🗸	😑 🖬 🕢 EmptyStat	tementNotInL 🛛 😥 EmptyWhileStmt, jaw	a 🛛 🕢 Empty Tr	yBlock, java 😰 🍡	•
inspection		1 packa	ge basic;			-
B 🔐 agovruletest			•			
😑 🥮 src/mair	Collete		class EmptyTryBlock (
			lic void bar() (
🖩 🔂 E		F4	try (
🖩 🔂 E	Show In	Alt+Shift+W) catch (Exception e) {			
		Ctrl+C	<pre>e.printStackTrace();</pre>			
	r 🔝 <u>C</u> opy 7 🌇 Copy Qualified Name	CII.1+C	3			
E D U	Paste	Ctrl+V	1			
🔅 🏭 braca	Easte	Delete				
🗈 🏭 contr		Defete	_			
🗈 🏭 desig	👲 Remove from Context	Ctrl+Alt+Shift+Down				
● 虚 impo ● 虚 javalo	Duliu Paul		•			
🛈 进 nami	Source	the strike w				
🛈 🏭 optim		Alt+Shift+T				
🗈 🏭 strict	🚵 Import		1			
🖬 🏭 string 🗈 🏭 unus	🛃 Esport		1			
		F5				
	🔗 Befresh	F5				
@# src/main @# src/test/ @# src/test/	, <mark>∻</mark> Befresh Dp <u>e</u> n Project Close Project	F5				
2# src/main -2# src/test/ 2# src/test/ @-34 JRE Sys	 Patrical Opgn Project Cloge Project Assign Working Sets 	F5				
- 🤐 src/main - 🥮 src/test/ - 🥮 src/test/ ⊛ - 🛋 JRE Sys ⊛ - 🚵 Maven D	 Patrical Opgn Project Cloge Project Assign Working Sets 					
2# src/main -2# src/test/ 2# src/test/ @-34 JRE Sys	 Perfection Opgin Project Cloge Project Assign Working Sets <u>B</u>un As <u>D</u>ebug As 					
- 🥮 src/main - 🥮 src/testy - 😅 src/testy - 🛋 JRE Sys - E 🚵 Maven D - E 📻 src	 Perfecth Opgn Project Cloge Project Assign Working Sets Bun As Debug As Profile As 			sla		2
- 29 src/main - 29 src/testy - 20 src/testy - 30 JRE Sys - 30 Maven D - 30 src - 20 target	Pefresh Opgn Project Opgn Project Osge Project Assign Working Sets Bun As Debug As Profile As Coyerage As		Violations Overview 🖾 Conso	le		2 2 2
- 29 src/main - 29 src/testy - 20 src/testy - 31 Sys - 3	A Pefresh Cpgn Project Cloge Project Cloge Project Assign Working Sets Bun As Debug As Profile As Coverage As Validate			le	Resource	\$ ~
- 29 src/main - 29 src/testy - 20 src/testy - 30 JRE Sys - 30 Maven D - 30 src - 20 target	A Pefresh Cpgn Project Cloge Project Cloge Project Assign Working Sets Bun As Debug As Profile As Coverage As Validate mat Maven		Notations Overview Consonings, Dothers	le	Resource	2 2 2 2 2
- 29 src/main - 29 src/testy - 20 src/testy - 30 JRE Sys - 30 Maven D - 30 src - 20 target	A Pefresh Cpgn Project Cloge Project Cloge Project Assign Working Sets Bun As Debug As Profile As Coverage As Validate		 Violations Overview © Consol nings, D others 5 litems) 19' 처럼 사용되지 않는 'method' 파라印 		UnusedFormalParameter,	java
- 🤐 src/main - 🤐 src/testy - 🤐 src/testy - 📾 JRE Sys - E Maven D - Src - E target - 🔛 pom, xm	Petresh Opgn Project Cloge Project design Working Sets Bun As Debug As Profile As Coverage As Validate Midate Tgam Compare With		 첫 Violations Overview E Consonings, D others Sitems)) / 처럼 사용되지 않는 'method' 파라미니지 않는 Private field T 가 선언되었음 	비터가 있을	UnusedFormalParameter, UnusedPrivateField, java	
- 28 src/main - 28 src/testy - 28 src/testy - 31 JRE Sys - 32 Maven D - 35 src -	Petresh Opgn Project Cloge Project design Working Sets Bun As Debug As Profile As Coverage As Validate Midate Tgam Compare With		 Definitions Overview ■ Consol nings, Dothers 5 Items) 19' 처럼 사용되지 않는 'method' 파라다 지 않는 Private field 'T'가 선언되었을 1지 않는 Private Method 'too()' 가 선언되었을 	비터가 있을	UnusedFormalParameter, UnusedPrivateField, java UnusedPrivateMethod, java	
- 28 src/main 28 src/testy 28 src/testy 29 src/testy 20 src 20 s	Perresh Opgn Project Opgn Project Osge Project Assign Working Sets Bun As Debug As Profile As Coyerage As Validate Maven Tgam Compare With Restare from Local History, Spring Tools Web Development Tools		 Initiations Overview ■ Consol nings, D others Sitems) 가 처럼 사용되지 않는 'method' 파라이 지 않는 Private field T 가 선언되었을 지 않는 Private Method 'too()' 가 선언! 에서 Assign된 변수 T 를 Final로 선언적 	비터가 있음 되었음 하지 않았음	UnusedFormalParameter,j UnusedPrivateField,java UnusedPrivateMethod,java UnusedPrivateField,java	a
- 28 src/main - 28 src/testy - 28 src/testy - 31 JRE Sys - 32 Maven D - 35 src -	Perresh Opgn Project Opgn Project Oggn Vorking Sets <u>Bun As Debug As Profile As Coyerage As Validate Maven Tgam Compare With Restore from Local History, Spring Tools Web Development Tools PDE Tools </u>		 첫 Violations Overview	미터가 있음 되었음 Sigh(2) 등 사용하는 2) 등필요함	UnusedFormalParameter, UnusedPrivateField, java UnusedPrivateMethod, jav UnusedPrivateField, java InefficientEmptyStringCher StringToString, java	a :ck,java
- 28 src/main 28 src/testy 28 src/testy 29 src/testy 20 src 20 s	Performer Opgin Project Opgin Project Opgin Project Opgin Project design Working Sets Bun As Debug As Debug As Profile As Covyerage As Validate Maven Tgam Compare With Restare fram Local History, Spring Tools Web Development Tools PDE Tools Aspect Tools		값 Violations Overview ☑ Consol nings, 0 others ☑ 5 items) ½ ½ 처럼 사용되지 않는 'method' 파라디 지 않는 Private field T 가 선언되었음 1지 않는 Private Method 'too()' 가 선언되었음 1지 않는 Private Method 'too()' 가 선언되었음 1지 않는 Private Method 'too()' 가 선언되었음 1지 않는 String, 한 프라이 와이 String, trim(), le 핵체에서 toString() 한수를 사용하는 것은	미터가 있음 되었음 51% 양았음 51% 2 사용하는 2 물필요한 2 사용할 필요 324	UnusedFormalParameter, UnusedPrivateField, java UnusedPrivateMethod, javi UnusedPrivateField, java InefficientEmptyStringCher StringToString, java UsalessStringValuaOf, java	a ck,java
- 28 src/main 28 src/testy 28 src/testy 29 src/testy 20 src 20 s	Perresh Opgn Project Opgn Project Oggn Vorking Sets <u>Bun As Debug As Profile As Coyerage As Validate Maven Tgam Compare With Restore from Local History, Spring Tools Web Development Tools PDE Tools </u>		중 Violations Overview Consol nings, 0 others Consol 5 litems) Y 처럼 사용되지 않는 'method' 파라디 지 않는 Private field T 가 선언되었음 1지 않는 Private Method 'too()' 가 선언되었음 N 성원 Y 1지 않는 Private Method 'too()' 가 선언되었음 N 성원 Y 1지 않는 Private Method 'too()' 가 선언되었음 N 성원 Y 1지 않는 String() 한수를 사용하는 것은 N 성원 N 성원 2 Ø Generate reports N 성원	미터가 있음 되었음 51% 양았음 51% 2 사용하는 2 물필요한 2 사용할 필요 324	UnusedFormalParameter, UnusedPrivateField, java UnusedPrivateField, java InefficientEmptyStingChei StringToSting, java Useloss StringValue0f, java InefficientStringButtering, ja	a :ck,java =
- 28 src/main 28 src/testy 28 src/testy 29 src/testy 20 src 20 s	Performer Opgin Project Opgin Project Opgin Project Opgin Project design Working Sets Bun As Debug As Debug As Profile As Covyerage As Validate Maven Tgam Compare With Restare fram Local History, Spring Tools Web Development Tools PDE Tools Aspect Tools	 Alt+Enter	중 Violations Overview Consol nings, 0 others Sitems) 1/2 처럼 사용되지 않는 'method' 파라미 1/3 않는 Private Method 'roo()' 가 선명되었음 1/3 않는 Private Method 'roo()' 가 선명되었음 1/4 Assign된 변수 T'를 Final로 선명함 2 Ning 를 체크하기 위해 String.trim().te 핵제에서 toString()한수를 사용하는 것은 2 Ø Generate reports Ø Clear violations reviews	미터가 있음 되었음 bl (3) (3) (3) (3) (4) (3) (4) (4) (4) (4) (4) (4) (4) (4) (4) (4	UnusedFormalParameter, UnusedPrivateField, java UnusedPrivateField, java InefficientEmptyStingCher StringToString, java UselessStringValueOf, java InefficientStringButlering, ja	a eck,java e ave
- 28 src/main 28 src/testy 28 src/testy 29 src/testy 20 src 20 s	Performation Project Coge Project Coge Project design Working Sets Bun As Debug As Profile As Coyerage As Validate Maven Tgam Compare With Restare from Local History, Spring Tools Web Development Tools PDE Tools Aspect Tools PMD	 Alt+Enter	중 Violations Overview Consol nings, 0 others Consol 5 litems) 가 처럼 사용되지 않는 'method' 파라디 지 않는 Private Method 'roo()' 가 선명되었음 1지 않는 Private Method 'roo()' 가 선명되었음 2 이 Generate reports 2 Clear violations reviews 2 Find Suspect Cut And Paste	미터가 있음 되었음 bl (3) (3) (3) (3) (4) (3) (4) (4) (4) (4) (4) (4) (4) (4) (4) (4	UnusedFormalParameter, UnusedPrivateField, Java UnusedPrivateField, Java InefficientEmptyStingChe- StringToString, Java UsalossStringValueOt, Java InefficientStringBultering, Ja StringInstantiation, Java AvoidThrowingNullPointer	a eck,java a ava Exception.
- 28 src/main 28 src/testy 28 src/testy 29 src/testy 20 src 20 s	Performation Project Coge Project Coge Project design Working Sets Bun As Debug As Profile As Coyerage As Validate Maven Tgam Compare With Restare from Local History, Spring Tools Web Development Tools PDE Tools Aspect Tools PMD	 Alt+Enter	 했 Violations Overview E Consolinings, 0 others Sitems) 가 처럼 사용되지 않는 'method' 파라티지 않는 Private field T 가 선언되었을 지 않는 Private Method 'noo()' 가 선언되었을 지 않는 Private Method 'noo()' 가 선언되었을 이 제시 Assigned 변수 T'를 Final로 선언된 Sting 를 치크하기 위해 String,trim(),te 적지에서 toString() 압수를 사용하는 것은 I Generate reports Clear Molations reviews Find Suspect Cut And Paste, E Clear PMD Wolstone 	비터가 있을 되었을 하지 않았음 ength() 를 사용하는 은 물필요한 역 사용할 필요 당해 nate 하지 말 것 전	UnusedFormalParameter, UnusedPrivateField, java UnusedPrivateField, java InefficientEmptyStingCher StringToString, java UselessStringValueOf, java InefficientStringButlering, ja	a eck,java a ava Exception.

Confirm generated report

Multiple file formats are supported for per-file inspection report generation. CSV, HTML, TXT, and XML files can be created under the reports folder inside the project.



Reviewing reports

Reports display all the inspection results in a single file view. Following is an example of an HTML report file opened in a browser.

0 11 - 0 - R R & Dat + 2383 0- 2 - 8	0.0	4.0				
	- M -					
4 CP C #TEV. GAC Attivol space the govelet and heap taffand even them	_	× 🖬				
		PMD report				
Problems found						
File	Line	Problem				
1 promein/java/basic/EnglyCalchBlock.java.	11	내용에 없는 Catch Block에 문제				
2 arc/main/java/basic/EmptyPhaily/Slock.java	8	Thatyblock时 ellors)合				
3 encinein/ave/basic/Empty/Stint.jave	6	친비 구분에 사용을 피하도록 함				
4 src/insin/java/basic/EmptyStatementNotinLoop.java	10	물요없는 문장 이어 있음				
5 arc/mehr/jave/basic/EngrlyStatementNotinLoop.jave	14	물효했는 문장 이야 연용				
5 promangava/basic/EmptyTryBlock.java	6	내용에 있는 1y 불룩이 문제함				
7 Inc/mail/java/basic/Empty/Weiastin/java	5	전 Wiles 구분데 사용되었음.				
8 pro/maitr/java/basic/UmecessaryConversionTemporary.java	8	기존 데이티(plmitics type)를 String으로 변혼할 때 분분요한 당시 변환 작업을 대하도록 함				
9 arc/main/java/brace/WhileLcopel-AustLeeBraces.java	7.	중골호없이 사용된 while문의 사용은 피하라				
D arc/main/java/controversial/UnnecessaryPasentheoses.java	6	결호가 없어도 되는 상황에서 불물교한 결호를 사용할 경우 대치 메소드 호흡처럼 보여서 소스 프 도의 가득상을 떨어뜨릴 수 있음.				
I inc/main/java/design/AnshactClass/WitroutAbstractMethod.java	3	Abstract Classifiel에 Abstract Method가 흔져하지 않은				
2 arc/main/java/design/AssignmentFoNonFinalStatic java	4	atatic 분류의 만경하지않은 사용 개공영				
3 arc/main/java/decign/AvoidConstantsInterface.ava	3	Interface는 클레스의 behavior 율 구변하는 데메만 사용해야 함.				
4 pro/main/java/design/AvoidPeassioningParameters.java	6	'ter' 처럼 파리미티 값을 직접 변경하지 달것				
5 mc/main/java/design/AusidSynchronized MVethodLevel.jma	6	method 레벨의 xynchronization MCI block 레벨 xynchronization 을 사용하는 것이 비용적합				
5 mc/main/java/design/Equals/kul.java	T	nul 깊과 비교하기 위해 equals 방수를 사용하였음.				
7 src/main/ava/design/FinalFieldCouldBeStatic java	6	final field를 Static 모듈 연양하면 overhead 큰 술량 수 있죠.				
III arc/main/java/decign/immutableField.java	5	생성자해서 Assign면 면수 'x'를 Finel로 선전하지 않았음				
B pro/main/java/design/SimpleDateFormatNeedsLocale.java	8	SimpleDateFormat 인스탠스를 생성할때 Locale 을 지정하는 것이 바람적함				
 src/main/ava/design/Simplit/BooleanExpressions.iava. 	0	boolean 사용 시 볼볼요한 비로 연신을 피하도록 함				
8 src/main/java/design/SwitchStrdsShouldHaveDetauit.java	7	Switch 귀분에는 빈도시 detault labelin 있어야 함				
2 src/main/java/design/UncommentedEmpts/Vethod.java	6	된 Method에 우의을 주기빌것				
S src.inain/ava/importstmb/Dublicate/imports.lava	3	'design EqualsMull' import분야 응폭 선전 되었음				
9 pro/main/java/importstmt/importFiomSamePackage.java.	2	'importation, Duplicate Imports' import 문어 요독 성업 되었음				
5 pro/main/ava/importstmt/importFioinSamePackage.java	2	8일 패키지에 있을 때는 incort분을 사용할 물 요가 없을				
8 src/main/avarimportstmt/importFiomSamePackage.iava	3	8일 패키지에 있을 때는 import분을 사용할 필요가 없을				
7 src/main/seva/sevalogging/SystemPrintin java	11	System out print 21 AIBB.				
8 src/main/java/naming/Misicading/VariabieName.java.	6	Valiables that are not final should not contain underscores (avcept for underscores in standard prefix/buffod).				
9 pro/main/lava/homing/Midleoding/VariableName.iava	8	non-field 에 들어 InL 으로 시작할.				
		non-field 이 좋아 m. 으로 시장함.				

Fixing broken Korean fonts in reports

PMD's report files use UTF-8 encoding to generate Korean text, so if you see broken Korean fonts follow the guideline below.

- Web browser : opening HTML reports, and your encoding is set to Korean:
 - Switch to UTF-8 (Unicode).
 - Microsoft Internet Explorer:
 - From the menu bar, choose View > Encoding > Unicode (UTF-8)
 - Mozilla Firefox:
 - From the menu bar, choose View > Character encoding > Unicode (UTF-8)
 - Google Chrome:
 - Click the Customize and Control button to the right of the URL field, then choose Encoding > Unicode (UTF-8)
- Microsoft Excel: does not support CSV files in UTF-8
 - Open such CSV files with text editors such as Notepad, store in a different encoding other than UTF-8, then open
 - with Excel.
 - Use Notepad as below.
 - 1. Open the CSV file with Notepad, then choose File > Save as ... from the menu.
 - 2. In the dialog, choose All files in the file type drop-down menu.
 - 3. Choose ANSI or Unicode encoding, then save the file.
 - 4. Open the file with Excel.
- Hudson PMD Plugin tool has a bug of Hudson PMD Plugin used in reporting PMD using the remote CI server, Hudson and cannot recognize UTF-8-based Korean (or other Asian) rule set files.
 - Until the bug fix is patched, you need to use english rule set file in order to use the PMD plug-in.

Utilizing inspection report statistics

You can use the hudson PMD Plugin to utilize the statistical information in the inspection reports.

To search statistics on violations from Hudson, it is confirmed as following.

- In the Hudson Dashboard, choose a project to review
- Choose PMD Warning from the left menu

					9	
ark-dev-pmdte	est × #14 × PMD Wa	rnings				
DMI	D Resu	14				
FPH	D Resu	II.				
Mari	nings Tre	bne				
wan	ings inc	inu				
All Warn	ings	New Wa	rnings	Fixed Warnings		
В7		87		0		
Sum	mary					
Total	High Priority		Normal Priority	Low Prio	Low Priority	
87	87		0	0		
Deta	ils					
Package	es Files Types	Warning	js Details New			
Package	es Files Types	Total	s Details New Distribution			
Package Package set.testi	riles Types Ties Types	Total 43				
Package Package sot.tasti unit.bas	rs Files Types ne inspact ic	Total 43 B	Distribution			
Package Package set.tasti unit.bas unit.bra	rs Files Types ne inspect ip De	Total 43 B 1	Distribution			
Package Package set.tasti unit.bas unit.bra unit.com	rs Files Types le inspact ile ce troversial	Total 43 8 1 1	Distribution			
Package Package set.ta.sti unit.bas unit.bra unit.con unit.des	rs Files Types pe inspact ic ce troversial ian	Total 43 8 1 1 1 12	Distribution			
Package Package set.tasti unit.bas unit.bra unit.con unit.des unit.imp	rs Files Types pe inspact ic ce troversial ian	Total 43 8 1 1	Distribution			
Package Package set.tasti unit.bas unit.bra unit.con unit.des unit.imp	rs Files Types pe inspact ic co troversial ign vortstmt alonging	Total 43 8 1 1 12 4	Distribution			
Package Package set.tasti unit.bas unit.bas unit.des unit.des unit.imp unit.iav unit.av	rs Files Types pe inspact ic co troversial ign vortstmt alonging	Total 43 8 1 1 12 4 4 1	Distribution			
Package Package set.tasti unit.bas unit.bas unit.des unit.des unit.imp unit.iav unit.av	rs Files Types Pe inspact ic ce troversial icn vortstmt alonging ning imization	Total 43 8 1 1 12 4 1 3	Distribution			
Package Package set.tasti unit.bas unit.bas unit.das unit.das unit.imp unit.imp unit.imp unit.imp	rs Files Types Pe inspact ic ce troversial icn vortstmt alonging ning imization	Total 43 8 1 1 12 4 1 3 3 3	Distribution			

Hudson ver. 1.319

PMD result view

- Warnings Trend: statistical trend of code inspection warnings in the project
 - All Warnings: total number of warnings occurred in the project
 - New Warnings: number or warnings from the most recent build
 - Fixed Warnings: number of fixed warnings in the most recent build
- Summary: displays statistics sorted by priorities (High/Normal/Low)
- Detail: displays detailed statistics with tabbed items as below
 - Packages: statistics per packages
 - Files: statistics per files
 - Types: statistics per violation types .
 - Warnings: all warnings
 - Details: all warnings with detailed information
 - New: new or unfixed warnings with detailed information

Detailed statistics view

Hudson PMD Plugin's statistics can be used to measure individual or team performances as well as metrics for code quality improvement.

Packages

Displays per-package violations as below.

Package	Total	Distribution
sat.testinspect	43	
unit.basic	8	
unit.braca	l	
unit.controversial	l	
<u>unit.design</u>	12	
<u>unit.impartstmt</u>	4	
<u>unit.javalogging</u>	1	
<u>unit.naming</u>	3	
unit.optimization	3	
unit.strictex	Z	
unit.stringnbuffar	5	
unit.unusedcode	4	

Files

Displays per-file violations as below.

ackages	Files	Types	Warnings	Details	New	
File To					Total	Distribution
Abstracto	lassWith	outAbstra	ctMethod.jav	a	1	-
Assignme	ntToNonf	FinalStatic	.java		1	
AvoidArra	ayLoops.j	ауа			1	
AvoidCor	istants Int	erface.jav	/a		1	
AvoidRea	ssigningP	arameter	s.java		1	
AvoidSyn	chronized	dAtMethod	Level.java		1	
AvoidThr	owingNull	PointerEx	ception.java		1	
AvoidThr	owingRaw	Exception	nTypes.java		1	
Duplicate	Imports.j	ava			1	
EmptyCa	tchBlock.	java			1	
EmptyFin	allyBlock.	java –			1	
EmptyIfS	tmt.java				1	
EmntySta	tementN	otTol oon.i	ava		2	

Types Displays per-type violations as below..

ackages Files Types Warnings Deta	is Naw
Туре	Total Distribution
AbstractClassWithoutAbstractMethod	2
AssignmentToNonFinalStatic	2
AvoidArrayLoops	2
Avoi dC onstants Interface	1
AvoidReassigningParameters	2
AvoidSynchronizedAtMethodLevel	2
AvoidThrowingNullPointerException	2
AvoidThrowingRawExceptionTypes	2
D uplicate Imports	3
EmptyCatchBlock	2
EmptyFinallyBlock	2
EmptyIfStmt	2
EmotyStatementNotToLoon	a

Warnings

Displays all warnings as below.

File	Line	Priority	Туре	Category
LogicalInspectionTestCode.java	151	High	Use less String Value Of	PMD_for_Edipse_3.2.5
LogicalInspectionTestCode.java	158	High	${\tt AbstractClassWithoutAbstractMethod}$	PMD_for_Edipse_3.2.5
LogicalInspectionTestCode.java	139	High	InefficientEmptyStringCheck	PMD_for_Edipse_3.2.5
LogicalInspectionTestCode.java	145	High	InefficientStringBuffering	PMD_for_Edipse_3.2.5
LogicalInspectionTestCode.java	129	High	AvoidThrowingNullPointerException	PMD_for_Edipse_3.2.5
LogicalInspectionTestCode.java	135	High	StringToString	PMD_for_Edipse_3.2.5
LogicalInspectionTestCode.java	119	High	AvoidArrayLoops	PMD_for_Edipse_3.2.5
LogicalInspectionTestCode.java	125	High	AvoidThrowingRawExceptionTypes	PMD_for_Edipse_3.2.5
LogicalInspectionTestCode.java	110	High	Unnecessary WrapperObjectCreation	PMD_for_Edipse_3.2.5
LogicalInspectionTestCode.java	107	High	Unnecessary WrapperObjectCreation	PMD_for_Edipse_3.2.5
LogicalInspectionTestCode.java	100	High	SystemPrintln	PMD_for_Edipse_3.2.5
LogicalInspectionTestCode.java	84	High	Unnecessary Parentheses	PMD_for_Edipse_3.2.5
LogicalInspectionTestCode.java	77	High	Switch Stmts Should Have Default	PMD_for_Edipse_3.2.5

1) Comma Separated Value

2) Continuous integration

3) Line of Code