



ANDROID ▾

JAVA ▾

JVM LANGUAGES ▾

SOFTWARE DEVELOPMENT

AGILE

CAREER

COMMUNICATIONS

DEVOPS

META JCG ▾

Home » Java » Enterprise Java » Exploit better the results of Pmd, Findbugs and CheckStyle.

ABOUT DANE DENNIS



Exploit better the results of Pmd, Findbugs and CheckStyle.

Posted by: Dane Dennis in Enterprise Java November 29th, 2013

Many Java static analysis tools exist right there, each one focus on a specific area and has its advantages, we can enumerate:

- **Pmd** which is a static rule-set based Java source code analyzer that identifies potential problems like:
 - Possible bugs—Empty try/catch/finally/switch blocks.
 - Dead code—Unused local variables, parameters and private methods.
 - Empty if/while statements.
 - Overcomplicated expressions—Unnecessary if statements, for loops that could be while loops.
 - Suboptimal code—Wasteful String/StringBuffer usage.
- **FindBugs** which looks for bugs in Java code. It uses static analysis to identify hundreds of different potential types of errors in Java programs.
- **Checkstyle** defines a set of available modules, each of which provides rules checking with a configurable level of strictness (mandatory, optional...). Each rule can raise notifications, warnings, and errors.

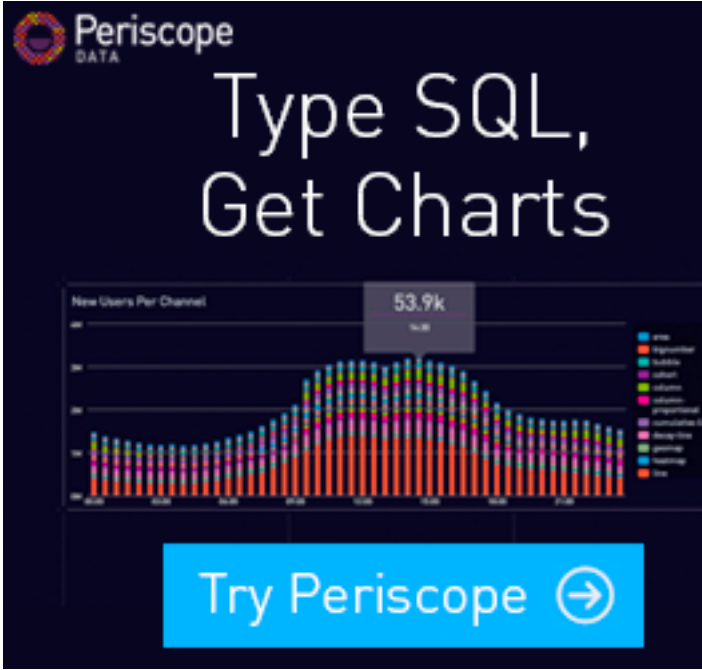
Many ways exist to exploit the results of these tools:

- **XML format:** XML files could be generated from each of these tools, and it can be used to create an HTML report or used by another tool to exploit the analysis result.
- **HTML format:** HTML report is the prefered way to generate reports and share them between the team, and you can create your custom report by using an xsl stylesheet.
- **IDE Plugins:** almost all known IDE provides plugins for these tools, which gives the possibility to discover all violations from the source code.

One of the problems with code quality tools is that they tend to overwhelm developers with problems that aren’t really problems — that is, false positives. When false positives occur, developers learn to ignore the output of the tool or abandon it altogether.

And to exploit better their result, it’s better to have a way to focus only on what we want and gives to developers a useful view. In this post we will discover another interesting way to exploit better the result of all known java static analysis tools, and query them like a database.

JArchitect and CQLinq



NEWSLETTER

159117 insiders are already enjoying weekly updates and complimentary whitepapers!
Join them now to gain **exclusive access** to the latest news in the Java world as well as insights about Android, Scala, Groovy and other related technologies.

Email address:

Sign up

JOIN US



With **1,240,600** monthly unique visitors and over **500** authors we are placed among the top related sites around. Constantly being on the lookout for partners; encourage you to join. So If you have a blog with unique and interesting content then you should check out our **JCG** partners program. You can be a **guest writer** for Java Code Geeks and hone your writing skills!

JArchitect is another static analysis tool which complements the other tools, it uses a code query language based on Linq (CQLinq) to query the code base like a database.

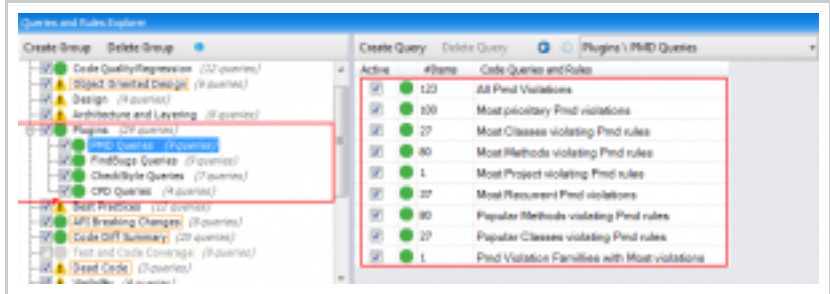
Until JArchitect 3 you can query only analysis data extracted from JArchitect, however the V4 gives the possibility to import the analysis result from many other static analysis tools and query them with CQLinq.

Let’s take as example the source code of the PDT core (the Php plugin for eclipse). and discover how we can exploit the analysis result of these tools from JArchitect.

To begin here are the steps to follows before requesting the analysis result:

- **Step1:** Analyse the project with PMD, CPD, FindBugs and CheckStyle. And generate the XML files containing the result.
- **Step2:** Analyze the project with JArchitect.
- **Step3:** Import all the xml files into JArchitect from the menu “Plugins->Import Plugins Result Files”

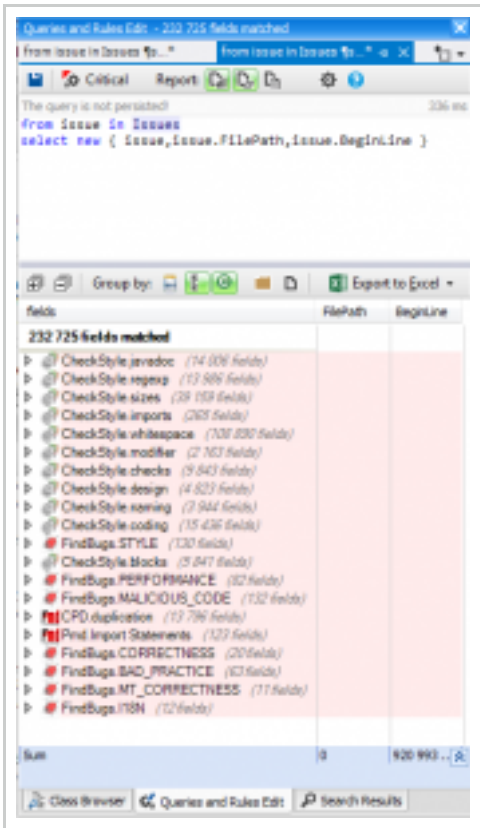
JArchitect provides by default many useful queries to request these tools, and all these queries could be customized easilly.



Let’s discover some CQLinq queries:

Get all issues:

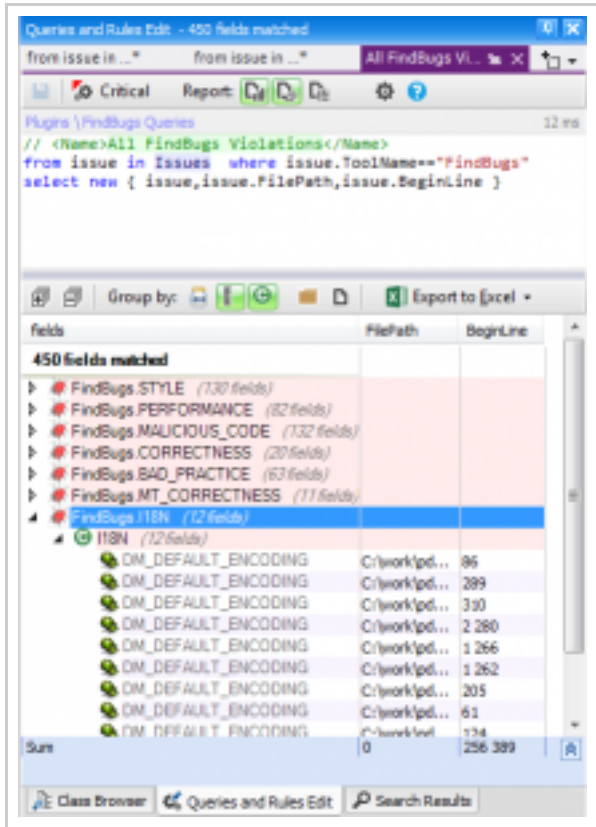
The request to get all issues is very simple, however as you can see it’s not very interesting, indeed it’s a challenge to exploit a result with 232 725 issues.



To exploit better the result of these tools we can filter it with CQLinq and focus only on what we want.

Request by tool

We can modify the first request and add a criteria about the tool concerned.



Request by ruleset

We can also filter by issue ruleset :

CAREER OPPORTUNITIES

Lower School Computer Science and Technology Teacher**La Jolla Country Day School**
La Jolla, CA
Mar, 26
Java Developer IIB**Computer Technologies International, Inc.**
Madison, WI
Mar, 18
Dell Software Sr.Solutions Principle Architect U Remote**Dell**
United States
Mar, 30
Dell Software Sr.Solutions Principle Architect U Remote**Dell**
Aliso Viejo, CA
Mar, 30
Dell Software Sr.Solutions Principle Architect U Remote**Dell**
Round Rock, TX
Mar, 30
Software Developer I, II and III (Java)**Steel I Solutions LLC**
Annapolis Junction, MD
Feb, 10
Android Framework Java Performance Engineer Santa Clara, CA**OSI Engineering, Inc.**
Santa Clara, CA
Apr, 01
Software Engineer**Google**
Madison, WI
Mar, 24
Need to Java, J2EE Developer,Columbus, OH, full time role**Nityo Infotech Services Pvt. Ltd.**
Columbus, OH
Mar, 28
Junior Software Developer**By Light Professional IT Services**
Remote
Mar, 24

1 2 ... 7132 »

☐ Freelance

☐ Full-time

☐ Internship

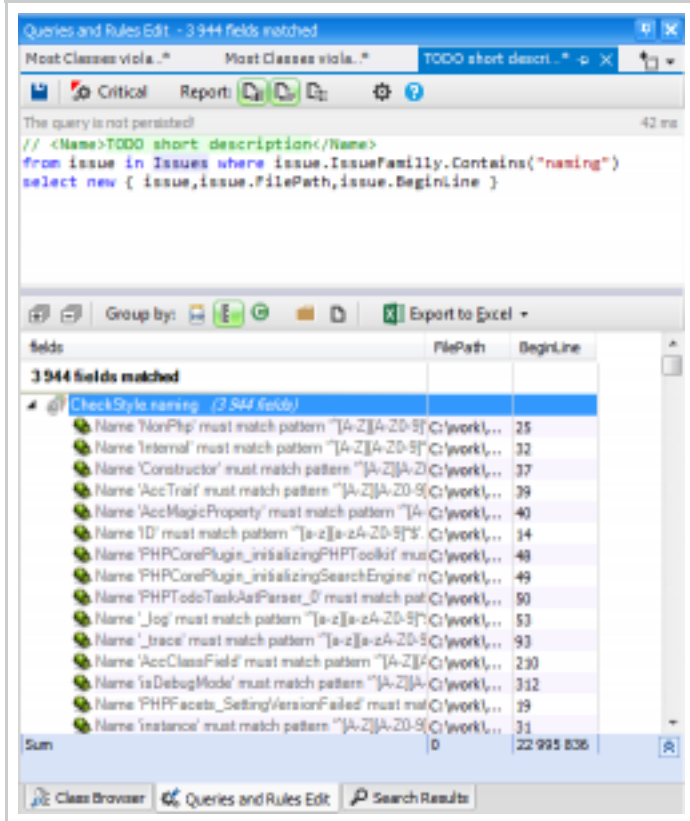
☐ Part-time

Keyword ...

Location ...

Country ...

Filter Results

jobs by **indeed**

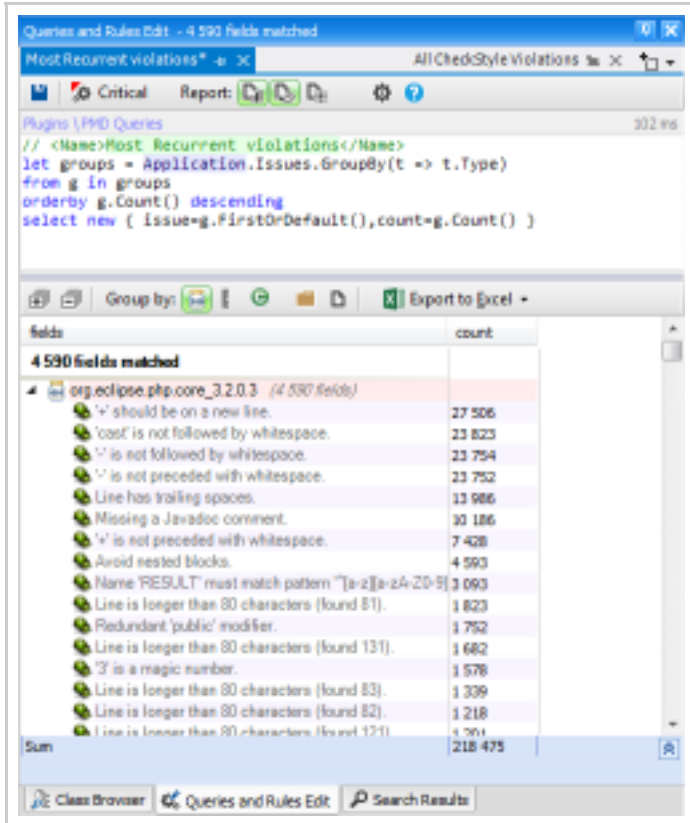
fields	FilePath	BeginLine
3944 fields matched		
[CheckStyle naming] (2544 fields)		
Name 'NonPip' must match pattern '[A-Z][A-Z0-9]*C\work\...		25
Name 'Internal' must match pattern '[A-Z][A-Z0-9]*C\work\...		32
Name 'Constructor' must match pattern '[A-Z][A-Z0-9]*C\work\...		37
Name 'AccTrait' must match pattern '[A-Z][A-Z0-9]*C\work\...		39
Name 'AccMagicProperty' must match pattern '[A-Z][A-Z0-9]*C\work\...		40
Name 'ID' must match pattern '[a-z][a-z0-9]*S\work\...		14
Name 'PHPCorePlugin_initializePHPToolkit' must match patC\work\...		48
Name 'PHPCorePlugin_initializeSearchEngine' must match patC\work\...		49
Name 'PHPTodoTaskAsParser_0' must match patC\work\...		50
Name 'Jag' must match pattern '[a-z][a-z0-9]*C\work\...		53
Name 'Trace' must match pattern '[a-z][a-z0-9]*C\work\...		93
Name 'AccClassField' must match pattern '[A-Z][A-Z0-9]*C\work\...		230
Name 'isDebugEnabled' must match pattern '[A-Z][A-Z0-9]*C\work\...		312
Name 'PHPFacets_SettingVersionFailed' must match patC\work\...		39
Name 'Instance' must match pattern '[A-Z][A-Z0-9]*C\work\...		31
Sum	0	22 995 836

Request by priority

We can also filter by priority:

Most recurrent issues

It's interesting to know which issues are the most reported by these tools.



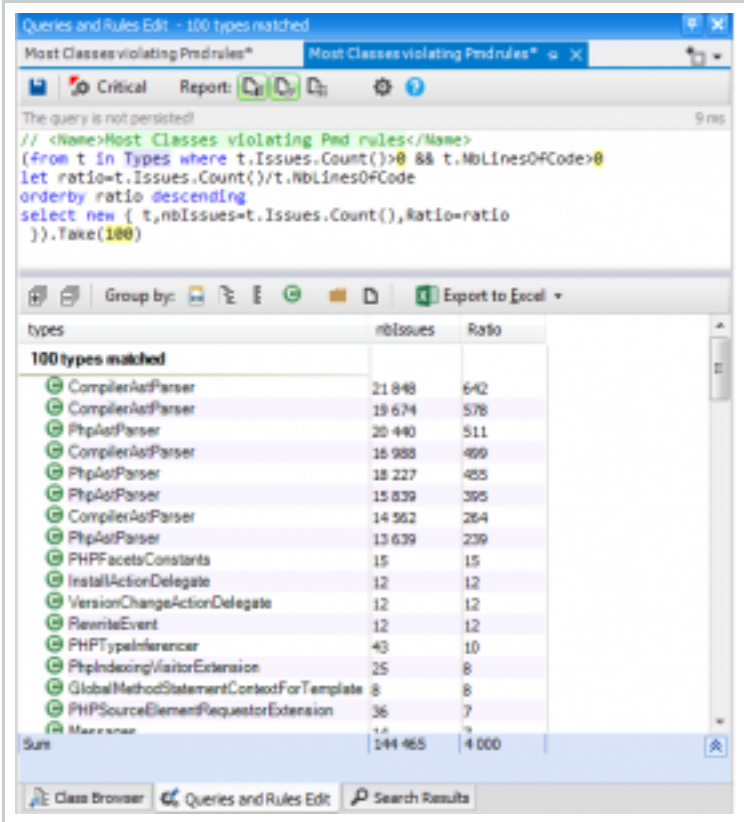
fields	count
4590 fields matched	
[org.eclipse.php.core_3.2.0.3] (4590 fields)	
'/' should be on a new line.	27 506
'/' is not followed by whitespace.	23 823
'/' is not followed by whitespace.	23 754
'/' is not preceded by whitespace.	23 752
Line has trailing spaces.	13 986
Missing a Javadoc comment.	10 186
'/' is not preceded with whitespace.	7 428
Avoid nested blocks.	4 593
Name 'RESULT' must match pattern '[a-z][a-z0-9]*C\work\...	3 063
Line is longer than 80 characters (found 81).	1 823
Redundant 'public' modifier.	1 752
Line is longer than 80 characters (found 131).	1 682
'3' is a magic number.	1 578
Line is longer than 80 characters (found 83).	1 309
Line is longer than 80 characters (found 82).	1 218
Line is longer than 80 characters (found 131).	1 104
Sum	218 475

Classes having most issues

It's very interesting to know the classes which contains many violations

As we can observe CheckStyle report thousand of issues and many of them could be ignored.

The previous query is interesting, but it's not give us exactly the classes with lack of quality, another useful metric to take into account is the NBLinesOfCode, it's normal that a class with many lines of code contains many issues, for that we can modify the previous request to calculate the ratio between the Issues count and the NBLinesofCode.



types	nbIssues	Ratio
100 types matched		
CompilerAsParser	21 848	642
CompilerAsParser	19 674	578
PhpAsParser	20 440	511
CompilerAsParser	16 988	499
PhpAsParser	18 227	455
PhpAsParser	15 839	395
CompilerAsParser	14 562	264
PhpAsParser	13 639	239
PHPFacetsConstants	15	15
InstallActionDelegate	12	12
VersionChangeActionDelegate	12	12
Event	12	12
PHPTypesReference	43	10
PhpIndexingVisitorExtension	25	8
GlobalMethodStatementContextForTemplate	8	8
PHPSourceElementRequestorExtension	36	7
Sum	144 465	4 000

What's very strange in this result is that the ratio of the 8 first classes is more than 200, in this case we have more than 200 issues by code line.

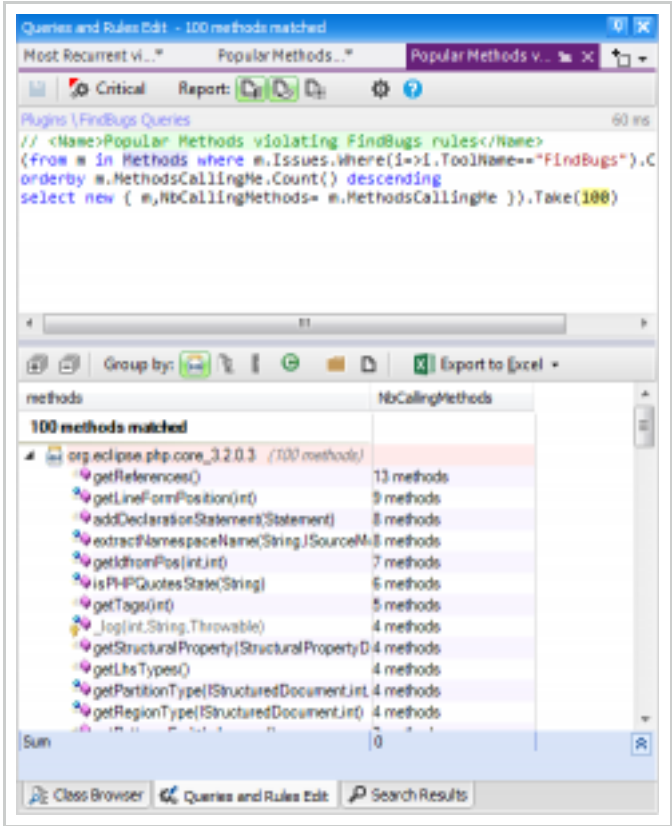
To explain this behavior let’s take a look at some lines of the CompilerAstParser:

The NbLinesOfCode is the number of statements and not the physical lines, and this Class declare many arrays , each one is declared by thousand of physical lines, however each array declaration is considered as one statement.

And as shown before for the most recurrent issues query, the following rule **`+’ should be on a new line.** is violated thousand of times for each array. Maybe it’s better to remove these kind of rules from the CheckStyle configuration file.

Most popular methods having issues

When the static analysis tools report the issues, it’s useful to locate which the priority issues to resolve? specially if it concerns bugs. It’s true that a bug could exist in a specific method, but what interesting to know is how many methods are impacted by the bug, and the popular method are the most used ones and it’s better to resolve them quickly.

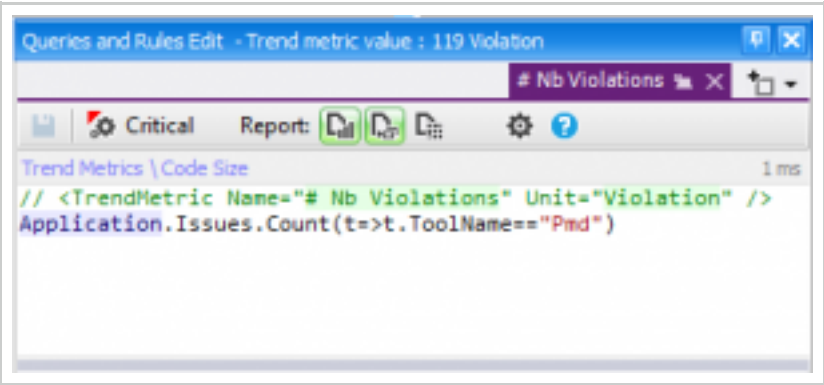


Using CQLinq we can combine the result of all these tools and also the result of JArchitect to create more elaborated queries, and add these checks to the build process.

Issues Trend

Having issues in a project is not an exception, we can say that’s normal, however we have to check the quality trend of the project. Indeed it’s a bad indicator if the number of issues grows after changes and evolutions. JArchitect provides the **Trend Monitoring** feature to create trend charts. Trend charts are made of trend metrics values logged over time at analysis time. More than 50 trend metrics are available per default and it is easy to create your own trend metrics.

Let’s create a trend metric for the Pmd issues:



And after you can easily create the trend chart to monitor the previous trend metric and add it to the JArchitect dashboard.

With this trend chart we can monitor the evolution of the Pmd issues, and try to understand the reasons when the metric grows over versions.

Customize the JArchitect report

JArchitect make possible to append extra report sections in the HTML report that lists some CQLinq queries. In the CQLinq Query Explorer panel, a particular CQLinq group reported is bordered with an orange rectangle.

You can also add to the report the Pmd trend chart:

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

☒

Sign me up for the newsletter!

Receive Email Notifications?

no, do not subscribe

Or, you can subscribe without commenting.

instantly

Post Comment

KNOWLEDGE BASE

- Courses
- Examples
- Resources
- Tutorials
- Whitepapers

PARTNERS

- Mkyong

THE CODE GEEKS NETWORK

- .NET Code Geeks
- Java Code Geeks
- System Code Geeks
- Web Code Geeks

HALL OF FAME

- “Android Full Application Tutorial” series
- 11 Online Learning websites that you should check out
- Advantages and Disadvantages of Cloud Computing – Cloud computing pros and cons
- Android Google Maps Tutorial
- Android JSON Parsing with Gson Tutorial
- Android Location Based Services Application – GPS location
- Android Quick Preferences Tutorial
- Difference between Comparator and Comparable in Java
- GWT 2 Spring 3 JPA 2 Hibernate 3.5 Tutorial
- Java Best Practices – Vector vs ArrayList vs HashSet

ABOUT JAVA CODE GEEKS

JCGs (Java Code Geeks) is an independent online community focused on creating the ultimate Java to Java developers resource center; targeted at the technical architect, technical team lead (senior developer), project manager and junior developers alike. JCGs serve the Java, SOA, Agile and Telecom communities with daily news written by domain experts, articles, tutorials, reviews, announcements, code snippets and open source projects.

DISCLAIMER

All trademarks and registered trademarks appearing on Java Code Geeks are the property of their respective owners. Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Examples Java Code Geeks is not connected to Oracle Corporation and is not sponsored by Oracle Corporation.