UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

Analisi statica del codice

Analizzatori di codice per Java

- **CheckStyle**
  - Scopre violazioni dello stile di programmazione
- **SpotBugs**
  - Scopre difetti nel codice
- PMD
  - Scopre problemi di incuria: es. codice non più usato, codice duplicato
- SonarQube
  - Include ed estende le funzionalità di CheckStyle, SpotBugs, PMD
  - Registra la storia dei controlli in un database
  - Modalità SaaS: SonarCloud

checkstyle

https://checkstyle.org/

- Scopre violazioni dello stile di programmazione
  - Sun Code Conventions (1999): https://www.oracle.com/technetwork/java/codeconvtoc-136057.html
  - Google Java Style Guide: https://google.github.io/styleguide/javaguide.html
- Plugin per IDE: Eclipse, IntelliJ IDEA, NetBeans, VS Code
- Plugin per build tool: Maven, **Gradle**,

  https://docs.gradle.org/current/userguide/checkstyle_plugin.html

Configurazione di Checkstyle

- Lista di controlli disponibili sul sito
  - http://checkstyle.sourceforge.net/checks.html
- Documento xml (es. **`checkstyle.xml`**)
  - specifica quali controlli effettuare
  - organizzato in moduli

Esempio:

```
<module name="Checker">
    <module name="JavadocPackage"/>
    <module name="TreeWalker">
        <module name="AvoidStarImport"/>
        <module name="ConstantName"/>
        <module name="EmptyBlock"/>
    </module>
</module>
```

Un esempio di report html

CheckStyle Audit

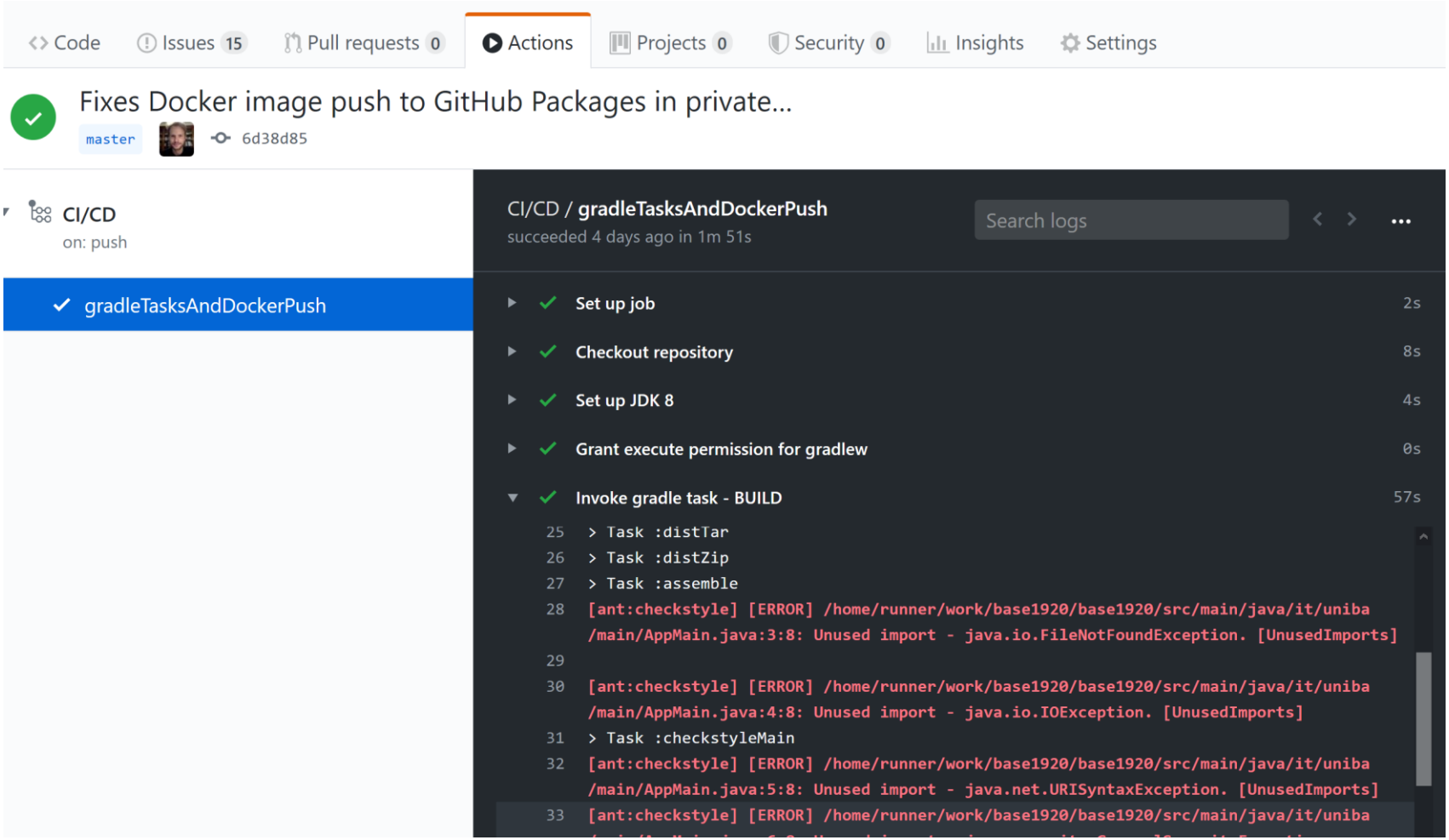
Designed for use with CheckStyle and Ant.

Summary	
Files	**Errors**
233	46

Files	
Name	**Errors**
source/net/sf/jomic/ui/JomicFrame.java	6
source/net/sf/jomic/tools/StringTools.java	3
source/net/sf/jomic/Jomic.java	2

File tests/net/sf/jomic/tools/TestTools.java	
Error Description	**Line**
Line is longer than 120 characters.	740
':' should be on a new line.	837

Un esempio di report nel log di deployment



SpotBugs

- Scopre difetti noti codice (400 bug patterns):
  - Bad practice, Correctness, Performance, Experimental, Internationalization, Malicious code vulnerability, Multithreaded correctness, Bogus random noise, Performance, Security, Dodgy code
- GUI
- Plugin per IDE: Eclipse, IntelliJ IDEA
- Plugin per build tool: Maven, **Gradle**

  https://spotbugs.readthedocs.io/en/latest/gradle.html

Bad practice

Esempio:

Nm: Field names should start with a lower case letter (NM_FIELD_NAMING_CONVENTION)

Names of fields that are not final should be in mixed case with a lowercase first letter and the first letters of subsequent words capitalized.

Correctness

Esempi:

NP: Read of unwritten field (NP_UNWRITTEN_FIELD)

The program is dereferencing a field that does not seem to ever have a non-null value written to it. Unless the field is initialized via some mechanism not seen by the analysis, dereferencing this value will generate a null pointer exception.

INT: Bad comparison of int value with long constant (INT_BAD_COMPARISON_WITH_INT_VALUE)

This code compares an int value with a long constant that is outside the range of values that can be represented as an int value. This comparison is vacuous and possibly incorrect.

Internationalization (I18N)

Esempio:

Dm: Consider using Locale parameterized version of invoked method (DM_CONVERT_CASE)

A String is being converted to upper or lowercase, using the platform's default encoding. This may result in improper conversions when used with international characters. Use the

- `String.toUpperCase( Locale l )`
- `String.toLowerCase( Locale l )`

versions instead.

Performance

Esempi:

UuF: Unused field (UUF_UNUSED_FIELD)

This field is never used. Consider removing it from the class.

IIL: NodeList.getLength() called in a loop (IIL_ELEMENTS_GET_LENGTH_IN_LOOP)

The method calls NodeList.getLength() inside the loop and NodeList was produced by getElementsByTagName call. This NodeList doesn't store its length, but computes it every time in not very optimal way. Consider storing the length to the variable before the loop.

Dodgy code (STYLE)

Esempi:

UCF: Useless control flow to next line (UCF_USELESS_CONTROL_FLOW_NEXT_LINE)

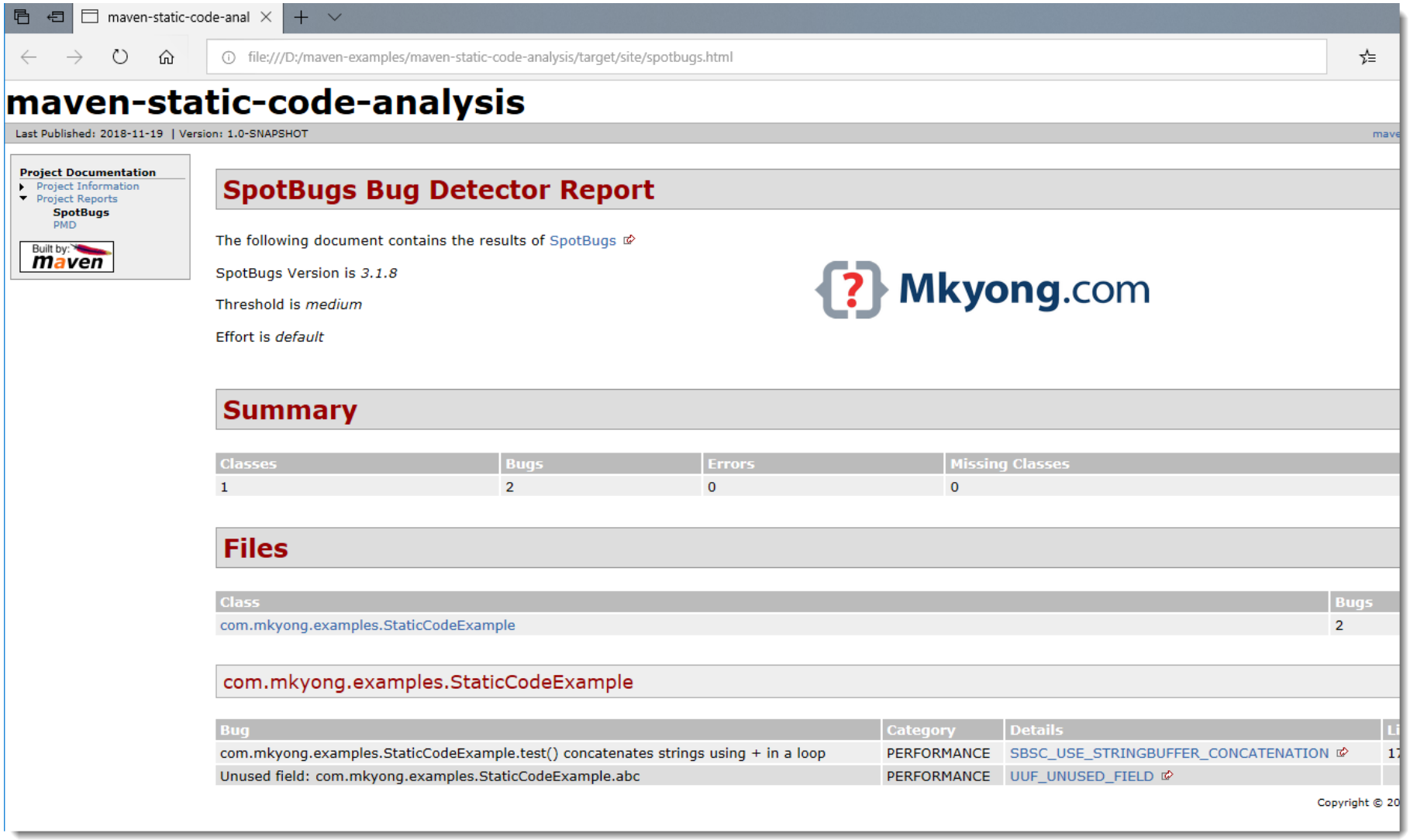
This method contains a useless control flow statement in which control flow follows to the same or following line regardless of whether or not the branch is taken. Often, this is caused by inadvertently using an empty statement as the body of an if statement, e.g.:

```
if (argv.length == 1);
 System.out.println("Hello, " + argv[0]);
```

DB: Method uses the same code for two branches (DB_DUPLICATE_BRANCHES)

This method uses the same code to implement two branches of a conditional branch. Check to ensure that this isn't a coding mistake.

Un esempio di report html



Un esempio di report nel log di deployment

```
59  > Task :spotbugsMain
60  Warning at xsl:variable on line 348 column 57 of default.xsl:
61    SXWN9001: A variable with no following sibling instructions has no effect
62  Warning at xsl:variable on line 351 column 57 of default.xsl:
63    SXWN9001: A variable with no following sibling instructions has no effect
64  SpotBugs rule violations were found. See the report at: file:///home/runner
    /work/base1920/base1920/build/reports/spotbugs/main.html
65
66  > Task :compileTestJava
67  > Task :processTestResources NO-SOURCE
68  > Task :testClasses
69  Warning at xsl:variable on line 348 column 57 of default.xsl:
70
71    SXWN9001: A variable with no following sibling instructions has no effect
72  Warning at xsl:variable on line 351 column 57 of default.xsl:
73    SXWN9001: A variable with no following sibling instructions has no effect
74  > Task :spotbugsTest
75  SpotBugs rule violations were found. See the report at: file:///home/runner
    /work/base1920/base1920/build/reports/spotbugs/test.html
```