**SonarQube Frequently asked questions (FAQ)**

**Is SonarQube Replacement for Checkstyle, PMD, FindBugs?**

Answers – YES and NO

**Why Yes?**

Sonar will run CheckStyle, FindBugs and PMD, as well as a few other “plugins” such as Cobertura (code coverage) by default for Java projects. The main added value, however, is that it stores the history in a database. You can then see the trend. Are you improving the code base or are you doing the opposite? Only a tool with memory can tell you that.

**Why No?**

SonarQube is currently on the way to deprecate PMD, Checkstyle and Findbugs and use their own technology to analyze Java code (called SonarJava). They do it, because they don’t want to spend their time fixing, upgrading (or waiting on it) those libraries (e.g. for Java 8), which for example uses outdated libraries. Well at least since SonarQube 6.3+ it seems to be that Findbugs is (at the moment) no longer supported as a plugin.

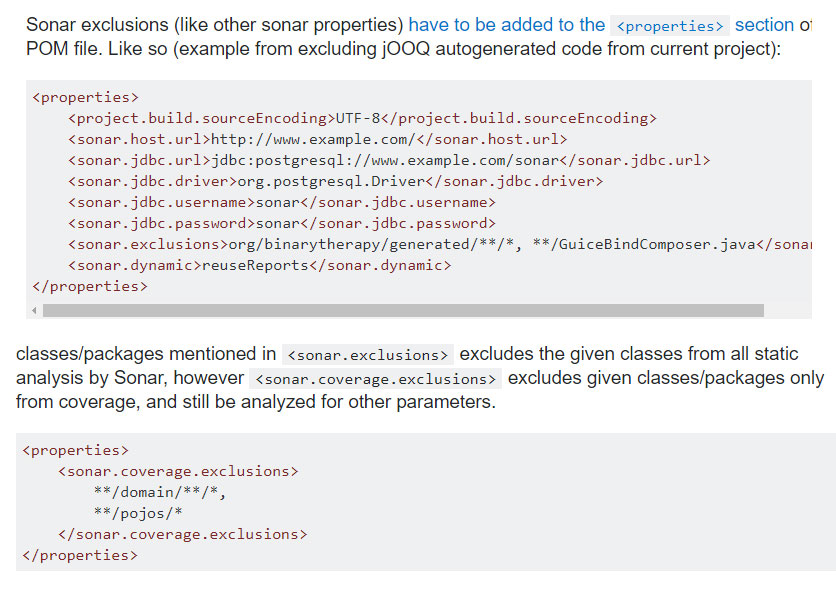
**What is the SonarQube Plugins available which can be integrated with Eclipse, IntelliJ IDEA, Visual Studio, Visual Studio Code, and Atom?**

Answer –

SonarLint is an extension to your favorite IDE that provides on-the-fly feedback to developers on new bugs and quality issues injected into their code.

**Where i can get the Support related to SonarQube?**

Configure Sonar to exclude files from Maven pom.xml



**Does Sonar support multiple language in same project ex. sonar.language=java,grvy?**

Starting with SonarQube 4.2, multi-language projects are supported.

This automatically happens when sonar.language is not set.

See sample project here: <https://github.com/bellingard/multi-language-project>

**How to exclude the directory in SonarQube?**

try something like this:

sonar.exclusions=src/java/test/\*\*

sonar.exclusions=system/\*\*, test/\*\*, application/third\_party/\*\*, application/logs/\*\*

**How to Increase SonarQube Heap Size?**

Whenever you get a issues like “SonarQube analysis failed java.lang.OutOfMemoryError: Java heap space”, you may following options to follow.

Option 1 – Reduce the analyze scope by either reducing the project size(split your project in smaller sub projects/modules) or reducing the set of rules that are analyzed.

Option 2 – Increase the memory size that can be consumed by the JVM.  This can be done by adding the following Environment Variable:

SONAR\_RUNNER\_OPTS=”-Xmx3062m -XX:MaxPermSize=512m -XX:ReservedCodeCacheSize=128m”

e.g Linux

> export SONAR\_SCANNER\_OPTS=”-Xmx3062m -XX:MaxPermSize=512m -XX:ReservedCodeCacheSize=128m”

e.g Windows

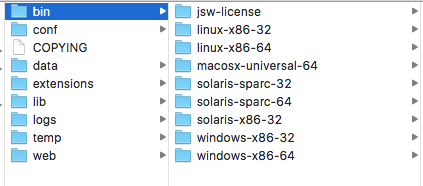
> set SONAR\_SCANNER\_OPTS=”-Xmx3062m -XX:MaxPermSize=512m -XX:ReservedCodeCacheSize=128m”

09. **SonarQube with Maven Tutorial – Code Quality for Java developers**

**Industry strength** code needs to statically & dynamically capture code quality. Also, more and more organizations are using “**production quality**” home assignments to shortlist candidates for job interviews. So, it really pays to set up code quality tools like SonarQube on your home development environment to get feedback on your code quality with the view to learm & improve.

This assumes that **Java 8** and **Maven 3** are set up.

**Step 1:** Download latest SonarQube from **http://www.sonarqube.org/downloads/**. This tutorial downloads SonarQube 5.3, which is a zip file, and unzip the file. The contents of unzipped **sonarqube-5.3 folder**:



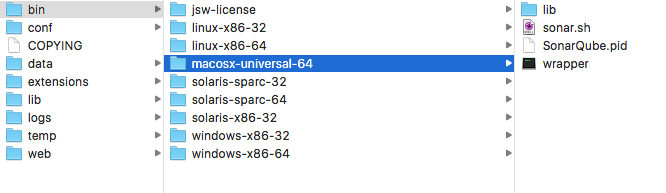
sonarqube-5.3 folder Contents

This tutorial uses “**macosx-universal-64**” for mac OS.

**Step 2:** **./sonar.sh** will start the sonar server on port number 9000. The usage is



|  |  |
| --- | --- |
| 1  2  3 | Usage: ./sonar.sh { console | start | stop | restart | status | dump } |



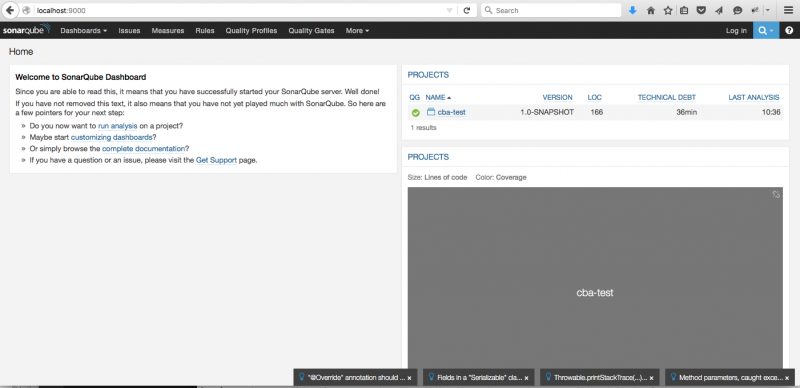
SonarQube MAC OSX

for this tutorial start SonarQube in your local machine in console mode



|  |  |
| --- | --- |
|  |  |

**Step 3**: After starting the SonarQube server, you can access it on a web browser by typing “**http://localhost:9000**“.



SonarQube Web Interface where Code Quality metrics are published

**Step 4:** Your maven projects can connect to the server running on localhost:9000 by making the following changes to your project’s **pom.xml** file



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65 | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">      <modelVersion>4.0.0</modelVersion>        <groupId>com.homeassigment</groupId>      <artifactId>homeassign-1-test</artifactId>      <version>1.0-SNAPSHOT</version>      <packaging>jar</packaging>        <name>homeassign-1-test</name>      <url>http://maven.apache.org</url>        <properties>          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>          <sonar.host.url>http://localhost:9000</sonar.host.url>          <logback.version>1.0.0</logback.version>          <slf4j.version>1.6.4</slf4j.version>      </properties>        <build>          <pluginManagement>              <plugins>                  <plugin>                      <groupId>org.codehaus.sonar</groupId>                      <artifactId>sonar-maven-plugin</artifactId>                      <version>5.3</version>                  </plugin>              </plugins>          </pluginManagement>      </build>        <dependencies>          <dependency>              <groupId>junit</groupId>              <artifactId>junit</artifactId>              <version>4.11</version>              <scope>test</scope>          </dependency>            <!-- Logging -->          <dependency>              <groupId>ch.qos.logback</groupId>              <artifactId>logback-classic</artifactId>              <version>${logback.version}</version>          </dependency>          <dependency>              <groupId>org.slf4j</groupId>              <artifactId>slf4j-api</artifactId>              <version>${slf4j.version}</version>          </dependency>          <dependency>              <groupId>org.slf4j</groupId>              <artifactId>log4j-over-slf4j</artifactId>              <version>${slf4j.version}</version>          </dependency>          <dependency>              <groupId>org.slf4j</groupId>              <artifactId>jcl-over-slf4j</artifactId>              <version>${slf4j.version}</version>          </dependency>        </dependencies>  </project> |

Key Points

**1)** sonar.host.url = http://localhost:9000

Can also be configured via Maven **settings.xml**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <profile>              <id>sonar</id>              <activation>                  <activeByDefault>true</activeByDefault>              </activation>              <properties> -->                  <!-- Optional URL to server. Default value is http://localhost:9000 -->                  <sonar.host.url>http://localhost:9000</sonar.host.url>              </properties>          </profile> |

**2)** Sonar & code quality hates System.out.println(…..) statements resulting as major code quality issue. So, use slf4j & logback in your code.

**3)** Latest “**Maven sonar plugin**” is used.



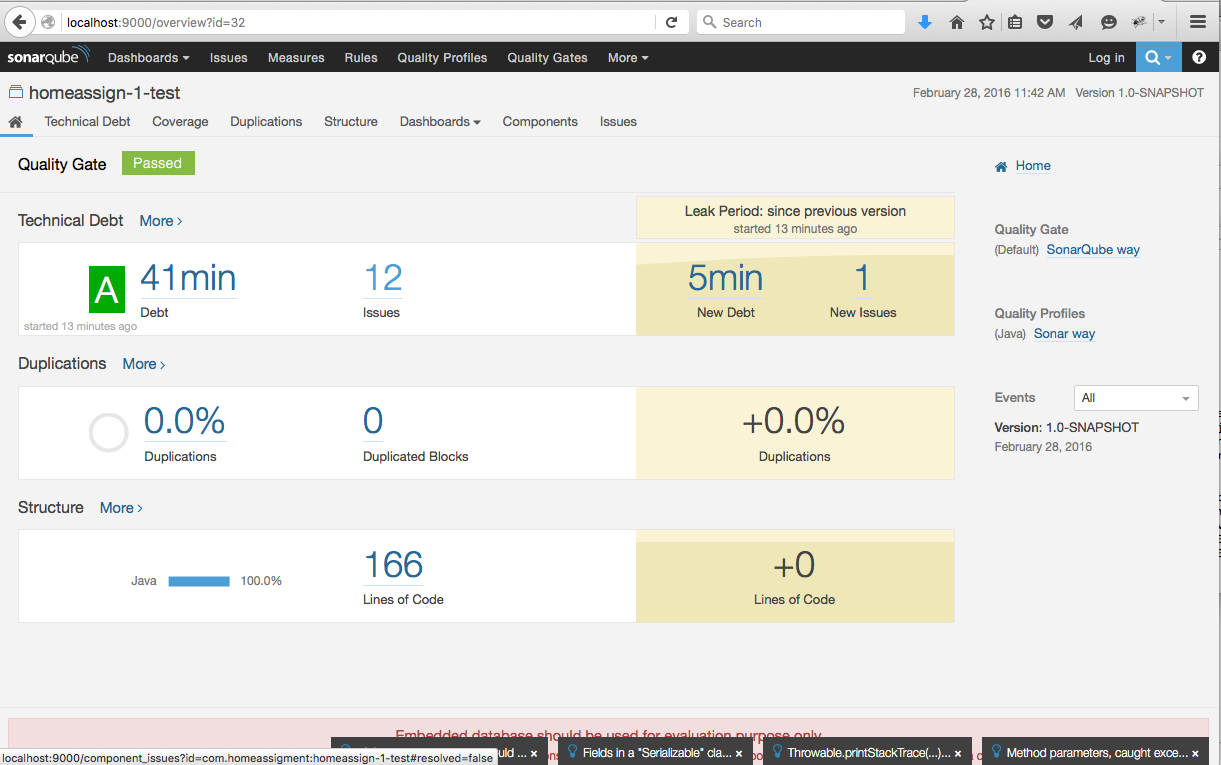
|  |  |
| --- | --- |
| 1  2  3  4  5 | <groupId>org.codehaus.sonar</groupId>     <artifactId>sonar-maven-plugin</artifactId>     <version>5.3</version> |

Step 5: You can publish the metrics with “mvn sonar:sonar”.



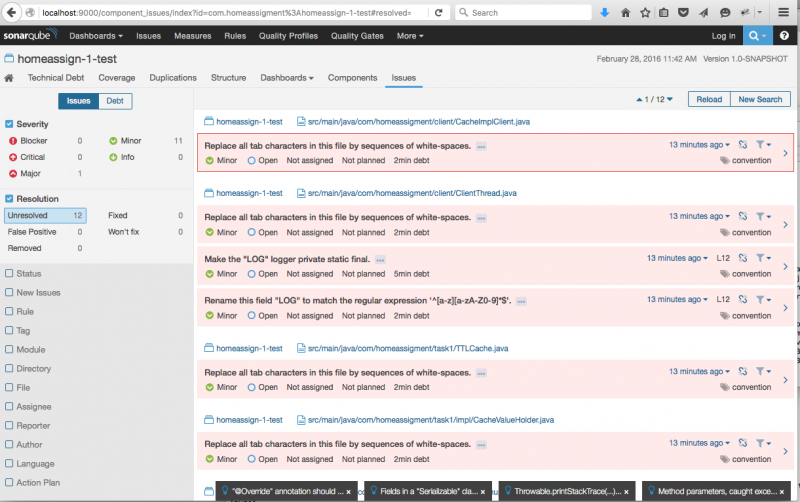
|  |  |
| --- | --- |
| 1  2  3 | homeassign-1-test arulk$ mvn clean install sonar:sonar |

Step 6: SonarQube Code metrics summary



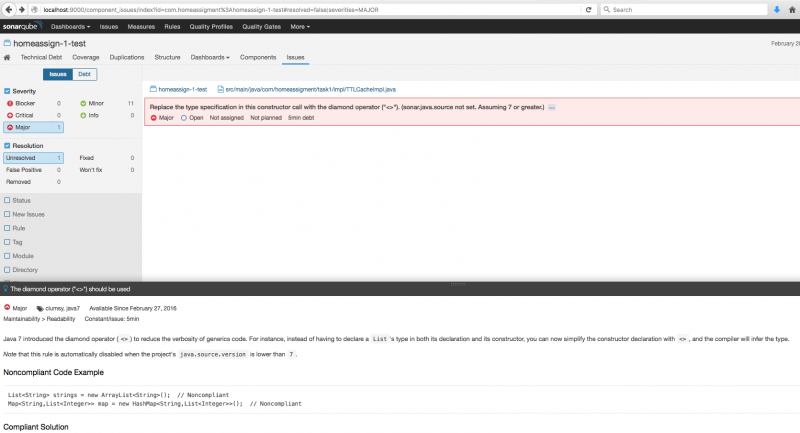
SonarQube Code Metrics Summary

You can drill down into the metrics for blocker, critical, and major issues.

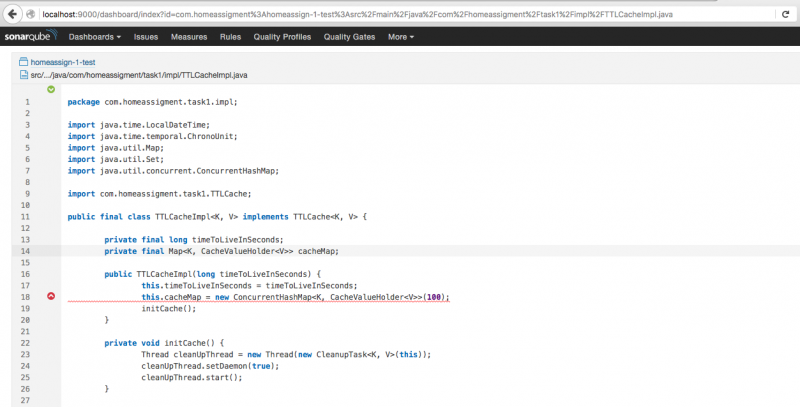


Code Quality Issues to fix

The above example has a major issue that requires attention:



Cause of major issue

You can drill down to code that has the issue:

Code with a major issue highlighted

So, get into the habit of passing your **home assignments**, **self-taught projects**, and **pre-interview assignments** via code quality tools like SonarQube. Every little thing matters to differentiate yourself from your competition.