Generate Codecoverage Report with Jacoco and Sonarqube

JaCoCo is a free code coverage library for Java, which has been created by the EclEmma team based on the lessons learned from using and integration existing libraries for many years.

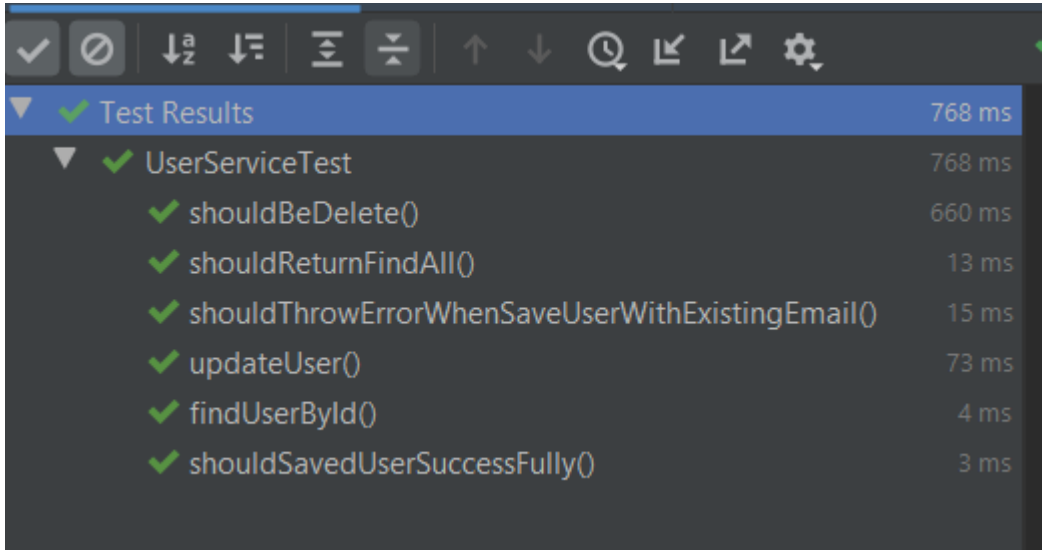
Prerequisite

- You have downloaded source code on this link.

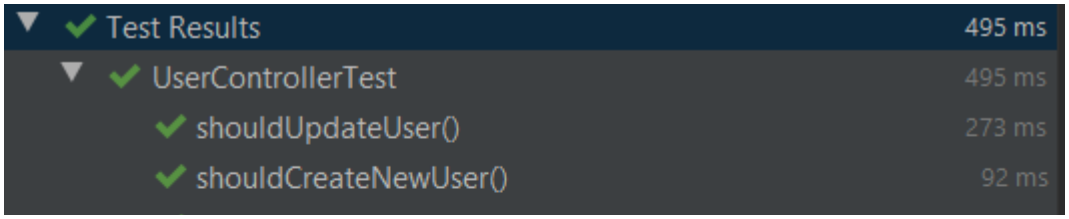
- You have download and setting Sonarqube and you can use community edition for this one.

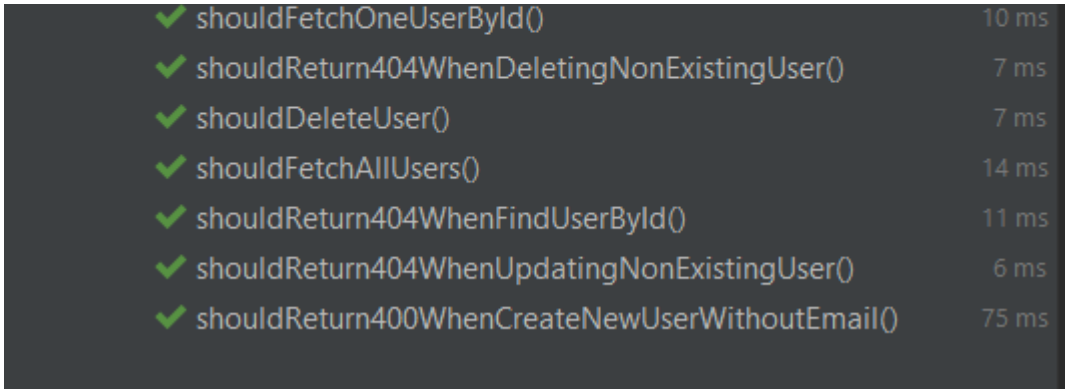
Check Before ..

Lets check again Unit Test that we have writed. Check on **UserServiceTest** and do Unit Test check.



and then, we check again **UserControllerTest.**

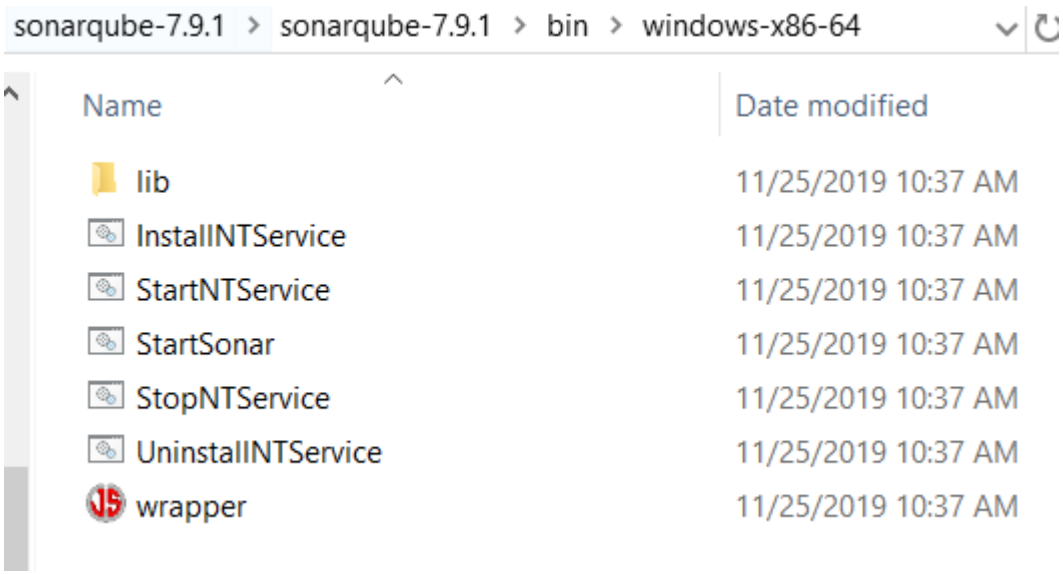




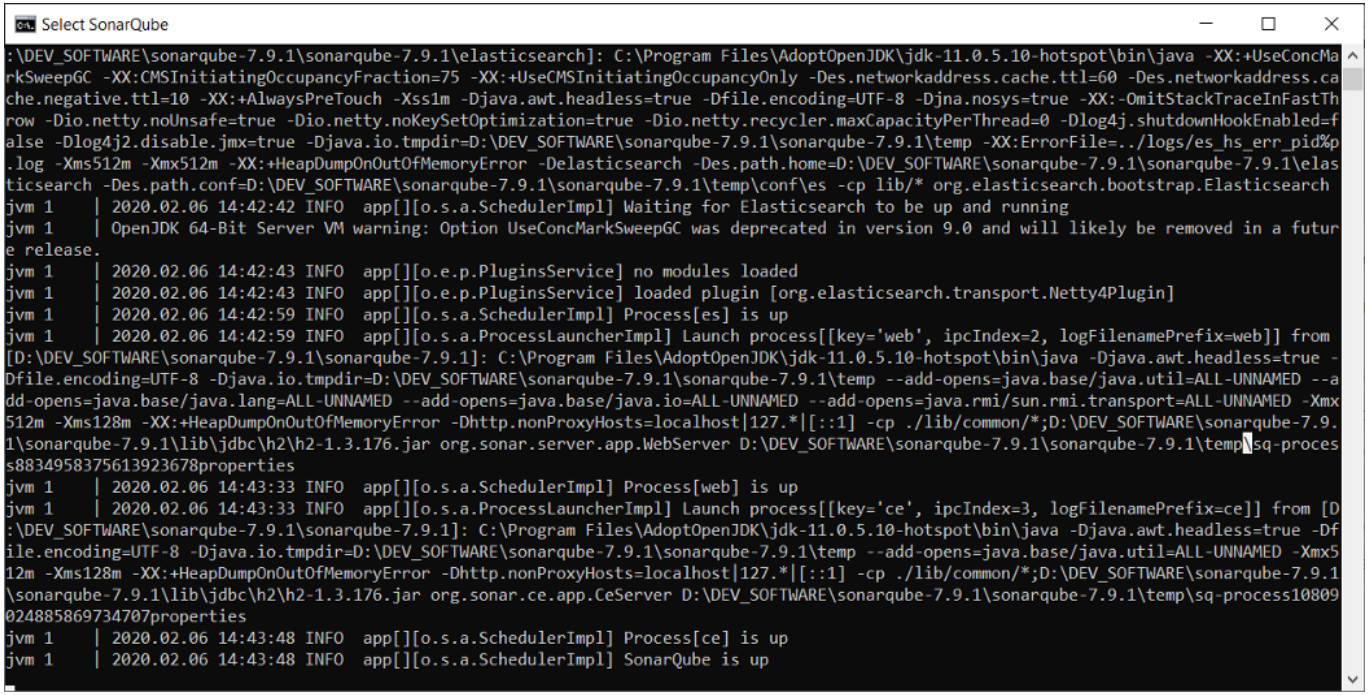
Okay, looks fine.

Start Sonarqube Server

Open your sonarqube directory and click **StartSonar.batch** (is depend on your operating system on your laptop)



wait for a moment until sonar server is running.



Generate Code Coverage Using Jacoco

- Add below Jacoco configuration on pom.xml properties section

```
<jacoco.version>0.8.3</jacoco.version>
<sonar.java.coveragePlugin>jacoco</sonar.java.coveragePlugin>
<sonar.dynamicAnalysis>reuseReports</sonar.dynamicAnalysis>
<sonar.jacoco.reportPath>${project.basedir}/../target/jacoco.exec</so
nar.jacoco.reportPath>
<sonar.language>java</sonar.language>
```

- and add also Jacoco plugin still on pom.xml under <plugin> section

```
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>${jacoco.version}</version>
    <configuration>
        <skip>${maven.test.skip}</skip>
        <destFile>${basedir}/target/coverage-reports/jacoco-
unit.exec</destFile>
        <dataFile>${basedir}/target/coverage-reports/jacoco-
unit.exec</dataFile>
        <output>file</output>
        <append>true</append>
        <excludes>
            <exclude>*MethodAccess</exclude>
        </excludes>
    </configuration>
    <executions>
        <execution>
            <id>jacoco-initialize</id>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
            <phase>test-compile</phase>
        </execution>
        <execution>
            <id>jacoco-site</id>
            <phase>verify</phase>
            <goals>
                <goal>report</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

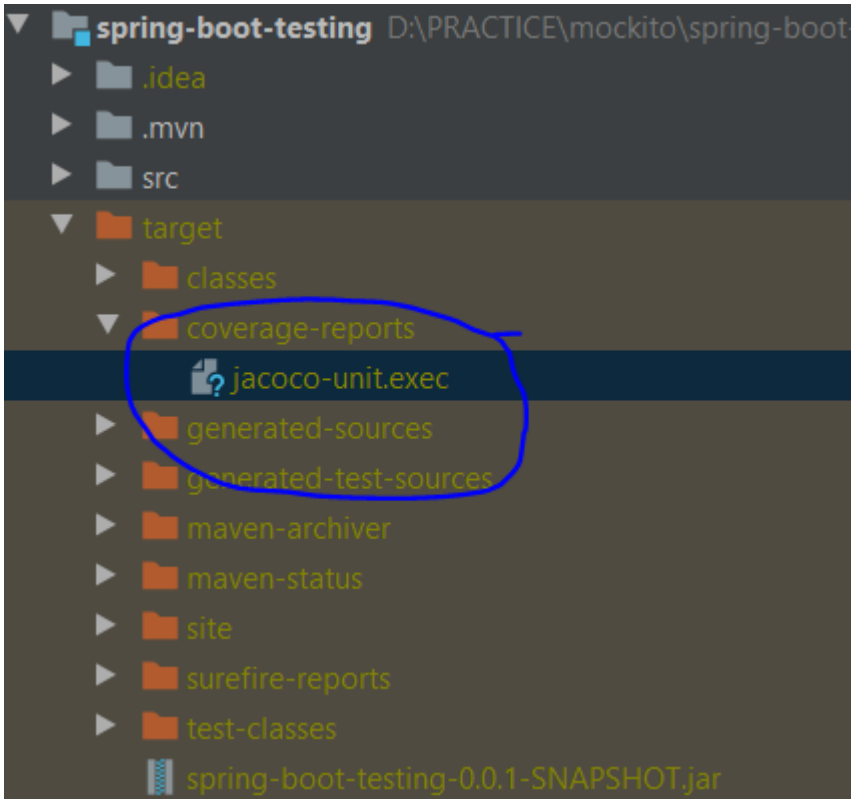
- after that, open **terminal** or **command prompt** and direct to root directory of project. Execute this maven command to build project.

mvn clean install

wait until build process has finished like this.

```
[INFO]
[INFO] --- maven-jar-plugin:3.1.2:jar (default-jar) @ spring-boot-testing ---
[INFO] Building jar: D:\PRACTICE\mockito\spring-boot-testing\target\spring-boot-testing-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.3:report (jacoco-site) @ spring-boot-testing ---
[INFO] Loading execution data file D:\PRACTICE\mockito\spring-boot-testing\target\coverage-reports\jacoco-unit.exec
[INFO] Analyzed bundle 'spring-boot-testing' with 6 classes
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ spring-boot-testing ---
[INFO] Installing D:\PRACTICE\mockito\spring-boot-testing\target\spring-boot-testing-0.0.1-SNAPSHOT.jar to C:\java_dev\libs\id\test\spring-boot-testing\0.0.1-SNAPSHOT\spring-boot-testing-0.0.1-SNAPSHOT.jar
[INFO] Installing D:\PRACTICE\mockito\spring-boot-testing\pom.xml to C:\java_dev\libs\id\test\spring-boot-testing\0.0.1-SNAPSHOT\spring-boot-testing-0.0.1-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  28.841 s
[INFO] Finished at: 2020-02-06T14:42:09+07:00
[INFO] ------------------------------------------------------------------------
```

Look, on **coverage report** under **target** folder there is file called **jacoco-unit.exec.** That file who used by Sonarqube to generate and display report about codecoverage, code quality , etc.



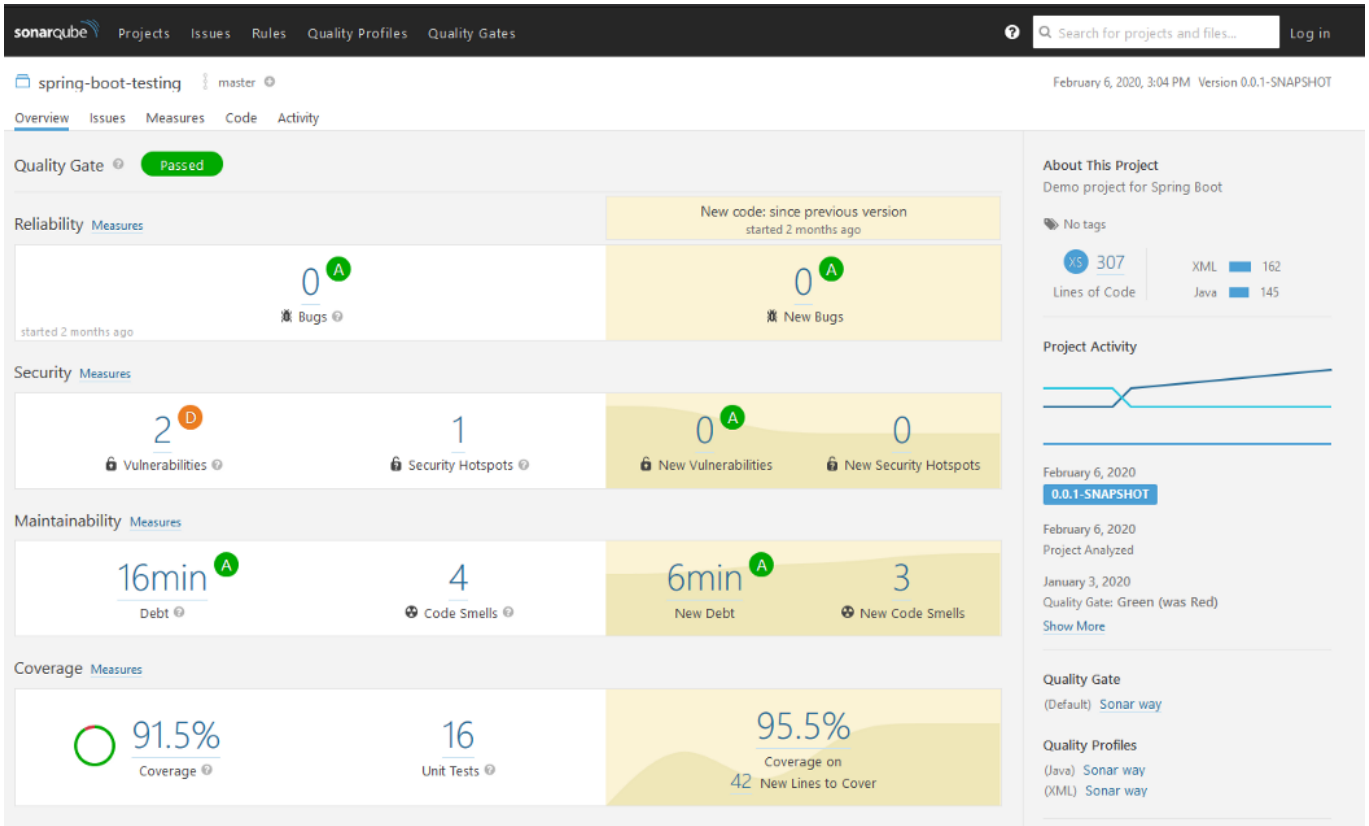
keep your terminal on root folder of project. and then execute this maven command to connect with Sonarqube.

mvn sonar:sonar

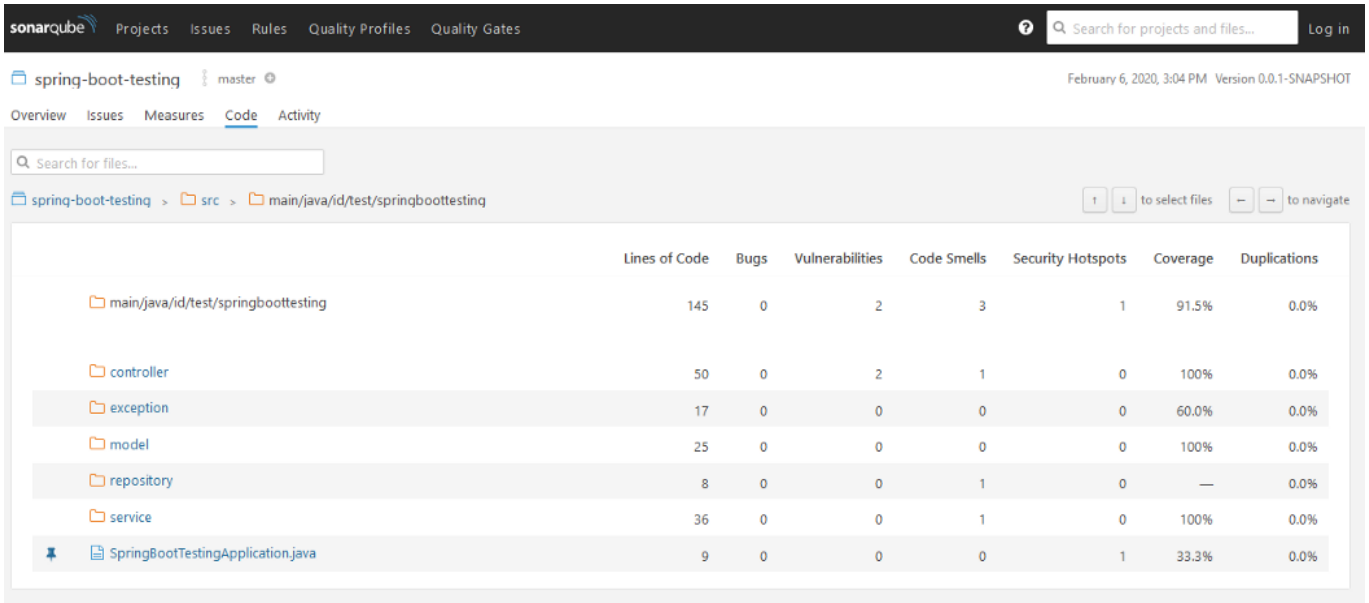
wait until build process has finished.

after finished, There are two link for you to open sonarqube on browser. click that link and automatically open your browser.

```
[INFO] 4 files had no CPD blocks
[INFO] Calculating CPD for 3 files
[INFO] CPD calculation finished
[INFO] Analysis report generated in 95ms, dir size=120 KB
[INFO] Analysis report compressed in 56ms, zip size=35 KB
[INFO] Analysis report uploaded in 21ms
[INFO] ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard?id=id.test%3Aspring-boot-testing
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] More about the report processing at http://localhost:9000/api/ce/task?id=AXAZh0CUja4xz_OuPKFC
[INFO] Analysis total time: 13.724 s
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  16.775 s
[INFO] Finished at: 2020-02-06T15:04:19+07:00
[INFO] ------------------------------------------------------------------------
```



this page is about report from out project. Seen that Coverage around 91.5%. Unit Test 16. and if you wanto to know detail of codecoverage from whole your code, click code tab.



	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
main/java/id/test/springboottesting	145	0	2	3	1	91.5%	0.0%
controller	50	0	2	1	0	100%	0.0%
exception	17	0	0	0	0	60.0%	0.0%
model	25	0	0	0	0	100%	0.0%
repository	8	0	0	1	0	—	0.0%
service	36	0	0	1	0	100%	0.0%
SpringBootTestingApplication.java	9	0	0	0	1	33.3%	0.0%

from that data, we can improve our code to be better again.

Conclusion

Jacoco and Sonarqube is important for many programmer. With both, we can improve quality of code little by little.