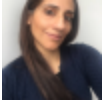


Exclusions from Jacoco Report

Last updated: June 4, 2021



Written by: Uzma Khan (<https://www.baeldung.com/author/uzmakhan>)

Gradle (<https://www.baeldung.com/category/gradle>)

Testing (<https://www.baeldung.com/category/testing>)

JaCoCo (<https://www.baeldung.com/tag/jacoco-library>)



End-to-end testing is a very useful method to make sure that your application works as intended. This highlights **issues in the overall functionality** of the software, that the unit and integration test stages may miss.

Playwright is an easy-to-use, but powerful tool that **automates end-to-end testing**, and supports all modern browsers and platforms.

When **coupled with LambdaTest** (an AI-powered cloud-based test execution platform) it can be further scaled to run the Playwright scripts in parallel across 3000+ browser and device combinations:

>> Automated End-to-End Testing With Playwright (</lambdatest-NPI-EA-2-a3yl>)

1. Introduction (/)

In this tutorial, we'll learn how to exclude certain classes and packages from JaCoCo (/jacoco) test coverage reports.


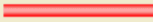
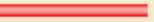








Generally, the candidates for exclusion can be configuration classes, POJOs, DTOs, as well as generated byte code. These carry no specific business logic, and it could be useful to exclude them from the reports in order to provide a better view of the test coverage.

We'll explore various ways of exclusion in both Maven and a Gradle project.

2. Example

Let's start with a sample project where we have all the required code already covered by tests.

Next, we'll generate the coverage report by running *mvn clean package* or *mvn jacoco:report*.

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 com.baeldung.domain		0%		0%	23	23	4	4	15	15	2	2
 com.baeldung.config		0%		n/a	2	2	2	2	2	2	1	1
 com.baeldung.dto		0%		n/a	2	2	2	2	2	2	2	2
 com.baeldung.generated		0%		n/a	2	2	2	2	2	2	1	1
 com.baeldung.service		87%		n/a	1	5	1	5	1	5	0	2
Total	177 of 191	7%	16 of 16	0%	30	34	11	15	22	26	6	8

(/wp-content/uploads/2021/06/Screenshot-2021-05-30-at-21.07.46.png)

This report shows that we already have the required coverage, and missed instructions should be excluded from JaCoCo report metrics.

3. Excluding Using Plugin Configuration

Classes and packages can be **excluded using standard * and ? wildcard syntax** in the plugin configuration:

- * matches zero or more characters
- ** matches zero or more directories
- ? matches a single character

3.1. Maven Configuration

Let's update the Maven plugin to add several excluded patterns:

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <configuration>
    <excludes>
      <exclude>com/baeldung/**/ExcludedPOJO.class</exclude>
      <exclude>com/baeldung/**/*DTO.*</exclude>
      <exclude>**/config/*</exclude>
    </excludes>
  </configuration>
  ...
</plugin>
```

Here we've specified the following exclusions:

- *ExcludedPOJO* class in any sub-package under *com.baeldung* package
- all classes with names ending in *DTO* in any sub-package under *com.baeldung* package
- the *config* package declared anywhere in the root or sub-packages

3.2. Gradle Configuration

We can also apply the same exclusions in a Gradle project.

First, we'll update the JaCoCo configuration in *build.gradle* and specify a list of exclusion, using the same patterns as earlier:

```

jacocoTestReport {
    //
    dependsOn test // tests are required to run before generating the
report

    afterEvaluate {
        classDirectories.setFrom(files(classDirectories.files.collect {
            fileTree(dir: it, exclude: [
                "com/baeldung/**/ExcludedPOJO.class",
                "com/baeldung/**/*DTO.*",
                "**/config/*"
            ])
        }))
    }
}

```

We use a closure to traverse the class directories and eliminate files that match a list of specified patterns. As a result, generating the report using `./gradlew jacocoTestReport` or `./gradlew clean test` will exclude all the specified classes and packages, as expected.

It's worth noting that the JaCoCo plugin is bound to the *test* phase here, which runs all the tests prior to generating the reports.

4. Excluding With Custom Annotation

Starting from JaCoCo 0.8.2, we can **exclude classes and methods by annotating them with a custom annotation** (`/java-custom-annotation`) with the following properties:

- The name of the annotation should include *Generated*.
- The retention policy of annotation should be *runtime* or *class*.

First, we'll create our annotation:

```

@Documented
@Retention(RUNTIME)
@Target({TYPE, METHOD, CONSTRUCTOR})
public @interface Generated {
}

```

Now we can annotate class(es) or method(s) or constructor(s) that should be excluded from the coverage report.

Let's use this annotation at the class level first:

```
@Generated
public class Customer {
    // everything in this class will be excluded from jacoco report
    because of @Generated
}
```

Similarly, we can apply this custom annotation to a specific method in a class:

```
public class CustomerService {

    @Generated
    public String getProductId() {
        // method excluded from coverage report
    }

    public String getCustomerName() {
        // method included in test coverage report
    }
}
```

Finally, let's apply the annotation to the constructor:

```
public class CustomerService {

    @Generated
    public CustomerService(){
        //constructor excluded from coverage report
    }

}
```

5. Excluding Lombok Generated Code


Project Lombok ([/intro-to-project-lombok](#)) is a popular library for greatly reducing boilerplate and repetitive code in Java projects.

Let's see how to **exclude all Lombok-generated bytecode by adding a property to *lombok.config* file** in our project's root directory:

```
lombok.addLombokGeneratedAnnotation = true
```

Basically, this property adds the *lombok.Generated* annotation to the relevant methods, classes, and fields of all the classes annotated with Lombok annotations, e.g. the *Product* class. As a result, JaCoCo then ignores all the constructs annotated with this annotation, and they're not shown in the reports.

Finally, we can see the report after applying all the exclusion techniques shown above:

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 com.baeldung.service	<div><div></div></div>	100%		n/a	0	4	0	4	0	4	0	2
Total	0 of 14	100%	0 of 0	n/a	0	4	0	4	0	4	0	2

([/wp-content/uploads/2021/06/Screenshot-2021-05-30-at-21.25.03.png](#))

6. Conclusion

In this article, we demonstrated various ways of specifying exclusions from the JaCoCo test report.

Initially, we excluded several files and packages using naming patterns in the plugin configuration. Then we saw how to use *@Generated* to exclude certain classes, as well as methods. Finally, we learned how to exclude all the Lombok-generated code from the test coverage report using a config file.

The code backing this article is available on GitHub. Once you're **logged in as a Baeldung Pro Member** ([/members/](#)), start learning and coding on the project.





Modern software architecture is often broken. Slow delivery leads to missed opportunities, innovation is stalled due to architectural complexities, and engineering resources are exceedingly expensive.

Orkes is the leading workflow orchestration platform built to enable teams to transform the way they develop, connect, and deploy applications, microservices, AI agents, and more.

With Orkes Conductor managed through Orkes Cloud, developers can focus on building mission critical applications without worrying about infrastructure maintenance to meet goals and, simply put, taking new products live faster and reducing total cost of ownership.

Try a 14-Day Free Trial of Orkes (/Orkes-NPI-EA-7) Conductor today.

COURSES

[ALL COURSES \(/COURSES/ALL-COURSES\)](#)

[BAELDUNG ALL ACCESS \(/COURSES/ALL-ACCESS\)](#)

[BAELDUNG ALL TEAM ACCESS \(/COURSES/ALL-ACCESS-TEAM\)](#)

[THE COURSES PLATFORM \(HTTPS://COURSES.BAELDUNG.COM\)](https://courses.baeldung.com)

SERIES

[JAVA "BACK TO BASICS" TUTORIAL \(/JAVA-TUTORIAL\)](#)

[LEARN SPRING BOOT SERIES \(/SPRING-BOOT\)](#)

[SPRING TUTORIAL \(/SPRING-TUTORIAL\)](#)

[GET STARTED WITH JAVA \(/GET-STARTED-WITH-JAVA-SERIES\)](#)

[SECURITY WITH SPRING \(/SECURITY-SPRING\)](#)

[REST WITH SPRING SERIES \(/REST-WITH-SPRING-SERIES\)](#)

[ALL ABOUT STRING IN JAVA \(/JAVA-STRING\)](#)

(/)

ABOUT

[ABOUT BAELDUNG \(/ABOUT\)](#)

[THE FULL ARCHIVE \(/FULL_ARCHIVE\)](#)

[EDITORS \(/EDITORS\)](#)

[OUR PARTNERS \(/PARTNERS/\)](#)

[PARTNER WITH BAELDUNG \(/PARTNERS/WORK-WITH-US\)](#)

[EBOOKS \(/LIBRARY/\)](#)

[FAQ \(/LIBRARY/FAQ\)](#)



[BAELDUNG PRO \(/MEMBERS/\)](#)

[TERMS OF SERVICE \(/TERMS-OF-SERVICE\)](#)

[PRIVACY POLICY \(/PRIVACY-POLICY\)](#)

[COMPANY INFO \(/BAELDUNG-COMPANY-INFO\)](#)

[CONTACT \(/CONTACT\)](#)

[PRIVACY MANAGER](#)