Petri Kainulainen

≡ Menu

If you want to save time by writing less test code, **take a look at my upcoming Test With Spring Course**.

Creating Code Coverage Reports for Unit and Integration Tests With the JaCoCo Maven Plugin

Petri Kainulainen August 17, 2013 110 comments

Code Coverage, Code Review, Integration Testing, Maven, Unit Testing

When I started using Java 7, I noticed right away that the Cobertura Maven plugin doesn't support it. This was a huge problem to me because I used code coverage reports every day.

I did some research and found the JaCoCo code coverage library. It looked interesting and I decided to give it a shot.

The problem was that configuring it was really hard and took a lot of time. I read numerous tutorials just to find out that the instructions given in them did not work for me. Then I ran into this blog post and everything fall into place.

Although that blog post was extremely valuable to me, it is a bit vague. I felt that a more detailed explanation about the usage of the JaCoCo Maven plugin would be valuable.

This blog post describes how we can create code coverage reports for unit and integration tests by using the JaCoCo Maven plugin.

The requirements of our build are following:

- Our build must create code coverage reports for both unit and integration tests when the tests are run.
- The code coverage reports must be created in separate directories. In other words, the code coverage report for unit tests must be created into a different directory than the code coverage report for integration tests.

Let's get started.

Note: The example application of this blog post is based on the example application of my blog post called Integration Testing with Maven. If you have not read it yet, I recommend that you read it before reading this blog post.

Configuring The JaCoCo Maven Plugin

We use the JaCoCo Maven plugin for two purposes:

1. It provides us an access to the JaCoCo runtime agent which records execution coverage data.
2. It creates code coverage reports from the execution data recorded by the JaCoCo runtime agent.

We can configure the JaCoCo Maven plugin by following these steps:

1. Add the JaCoCo Maven plugin to the *plugins* section of our POM file.

2. Configure the code coverage report for unit tests.
3. Configure the code coverage report for integration tests.

These steps are described with more details in the following.

Adding The JaCoCo Maven Plugin to The POM File

We can add the JaCoCo Maven plugin to our POM file by adding the following plugin declaration to its *plugins* section:

```
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.7.5.201505241946</version>
</plugin>
```

If you use Java 7 (or older), and the 0.7.x version is not working properly, downgrade to version 0.6.5.201403032054.

Let's move on and find out how we can configure the code coverage report for our unit tests.

Configuring The Code Coverage Report for Unit Tests

We can configure the code coverage report for unit tests by adding two executions to the plugin declaration. These executions are described in the following:

1. The first execution creates a property which points to the JaCoCo runtime agent. Ensure that the execution data is written to the file *target/coverage-reports/jacoco-ut.exec*. Set the name of the property to *surefireArgLine*. The value of this property is passed as a VM argument when our unit tests are run.
2. The second execution creates the code coverage report for unit tests after unit tests have been run. Ensure that the execution data is read from the file *target/coverage-reports/jacoco-ut.exec* and that the code coverage report is written to the directory *target/site/jacoco-ut*.

The relevant part of our plugin configuration looks as follows:

```
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.7.5.201505241946</version>
    <executions>
        <!--
            Prepares the property pointing to the JaCoCo runtime agent which
            is passed as VM argument when Maven the Surefire plugin is executed
        -->
        <execution>
            <id>pre-unit-test</id>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
            <configuration>
                <!-- Sets the path to the file which contains the execution dat
                <destFile>${project.build.directory}/coverage-reports/jacoco-ut
                <!--
                    Sets the name of the property containing the settings
                    for JaCoCo runtime agent.
                -->
                <propertyName>surefireArgLine</propertyName>
            </configuration>
        </execution>
        <!--
            Ensures that the code coverage report for unit tests is created aft
            unit tests have been run.
        -->
        <execution>
            <id>post-unit-test</id>
            <phase>test</phase>
            <goals>
                <goal>report</goal>
            </goals>
            <configuration>
                <!-- Sets the path to the file which contains the execution dat
                <dataFile>${project.build.directory}/coverage-reports/jacoco-ut
                <!-- Sets the output directory for the code coverage report. --
                <outputDirectory>${project.reporting.outputDirectory}/jacoco-ut
            </configuration>
        </execution>
    </executions>
</plugin>
```

Let’s find out how we can configure the code coverage report for our integration tests.

Configuring The Code Coverage Report for Integration Tests

We can configure the code coverage report for integration tests by adding two executions to the plugin declaration. These executions are described in the following:

1. This first execution creates a property which points to the JaCoCo runtime agent. Ensure that the execution data is written to the file *target/coverage-reports/jacoco-it.exec*. Set the name of the property to *failsafeArgLine*. The value of this property is passed as a VM argument when our integration tests are run.

2. Create an execution which creates the code coverage report for integration tests after integration tests have been run. Ensure that the execution data is read from the file *target/coverage-reports/jacoco-it.exec* and that the code coverage report is written to the directory *target/site/jacoco-it*.

The relevant part of our plugin configuration looks as follows:

```
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.7.5.201505241946</version>
    <executions>
        <!-- The Executions required by unit tests are omitted. -->
        <!--
            Prepares the property pointing to the JaCoCo runtime agent which
            is passed as VM argument when Maven the Failsafe plugin is executed
        -->
        <execution>
            <id>pre-integration-test</id>
            <phase>pre-integration-test</phase>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
            <configuration>
                <!-- Sets the path to the file which contains the execution dat
                <destFile>${project.build.directory}/coverage-reports/jacoco-it
                <!--
                    Sets the name of the property containing the settings
                    for JaCoCo runtime agent.
                -->
                <propertyName>failsafeArgLine</propertyName>
            </configuration>
        </execution>
        <!--
            Ensures that the code coverage report for integration tests after
            integration tests have been run.
        -->
        <execution>
            <id>post-integration-test</id>
            <phase>post-integration-test</phase>
            <goals>
                <goal>report</goal>
            </goals>
            <configuration>
                <!-- Sets the path to the file which contains the execution dat
                <dataFile>${project.build.directory}/coverage-reports/jacoco-it
                <!-- Sets the output directory for the code coverage report. --
                <outputDirectory>${project.reporting.outputDirectory}/jacoco-it
            </configuration>
        </execution>
    </executions>
</plugin>
```

That's it. We have now configured the JaCoCo Maven plugin. Our next step is to configure the Maven Surefire plugin. Let's find out how we can do this.

We use the Maven Surefire plugin to run the unit tests of our example application. Because we want to create a code coverage report for our unit tests, we have to ensure that the JaCoCo agent is running when our unit tests are run. We can ensure this by adding the value of the *surefireArgLine* property as the value of the *argLine* configuration parameter.

The configuration of the Maven Surefire plugin looks as follows (the required change is highlighted):

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.15</version>
    <configuration>
        <!-- Sets the VM argument line used when unit tests are run. -->
        <argLine>${surefireArgLine}</argLine>
        <!-- Skips unit tests if the value of skip.unit.tests property is true
        <skipTests>${skip.unit.tests}</skipTests>
        <!-- Excludes integration tests when unit tests are run. -->
        <excludes>
            <exclude>**/IT*.java</exclude>
        </excludes>
    </configuration>
</plugin>
```

We are almost done. The only thing left for us to do is to configure the Maven Failsafe plugin. Let’s find out how we can do it.

Configuring The Maven Failsafe Plugin

The integration tests of our example application are run by the Maven Failsafe plugin. Because we want to create a code coverage report for our integration tests, we have to ensure that the JaCoCo agent is running when our integration tests are run. We can do this by adding the value of the *failsafeArgLine* property as the value of the *argLine* configuration parameter.

The configuration of the Maven Failsafe plugin looks as follows (the required change is

```
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-failsafe-plugin</artifactId>
    <version>2.15</version>
    <executions>
        <!--
            Ensures that both integration-test and verify goals of the Failsafe
            plugin are executed.
        -->
        <execution>
            <id>integration-tests</id>
            <goals>
                <goal>integration-test</goal>
                <goal>verify</goal>
            </goals>
            <configuration>
                <!-- Sets the VM argument line used when integration tests are r
                <argLine>${failsafeArgLine}</argLine>
                <!--
                    Skips integration tests if the value of skip.integration.tes
                    is true
                -->
                <skipTests>${skip.integration.tests}</skipTests>
            </configuration>
        </execution>
    </executions>
</plugin>
```

Creating Code Coverage Reports

We have now successfully finished the required configuration. Let's see how we can create code coverage reports for unit and integration tests.

The example application of this blog post has three build profiles which are described in the following:

- The *dev* profile is used during development and it is the default profile of our build. When this profile is active, only unit tests are run.
- The *integration-test* profile is used to run integration tests.
- The *all-tests* profile is used to run both unit and integration tests.

We can create different code coverage reports by running the following commands at command prompt:

- The command *mvn clean test* runs unit tests and creates the code coverage report for unit tests to the directory *target/site/jacoco-ut*.

- The command *mvn clean verify -P integration-test* runs integration tests and creates the code coverage report for integration tests to the directory *target/site/jacoco-it*.
- The command *mvn clean verify -P all-tests* runs unit and integration tests and creates code coverage reports for unit and integration tests.

That's all for today. As always, the example application of this blog post is available at Github.

RELATED POSTS

UNCATEGORIZED /

My Favorite Blog Posts of 2015

WEEKLY /

Java Testing Weekly 6 / 2016

APACHE WICKET /

Mocking Spring Beans with Apache Wicket And Mockito

About the Author

Petri Kainulainen is passionate about software development and continuous improvement. He is specialized in software development with the Spring Framework and is the author of Spring Data book.

About Petri Kainulainen →

Connect With Me

106 comments… add one

Les Hazlewood

September 17, 2013, 23:13

Nice writeup!

Question: How do you fail the build if IT code coverage falls below a certain % ?

REPLY

Petri

September 17, 2013, 23:43

Thanks! You can use the *check* goal of the JaCoCo Maven plugin to fail the build if the code coverage is lower than a threshold value.

I haven't personally used it but I assume that you have to add a new execution which is run after the integration tests have been run (*post-integration-test* phase). I can try this out tomorrow (is quite late here and I have to sleep a bit before work).

REPLY

Stefan Brenner

September 25, 2013, 16:39

Nice article that helped me a lot to understand the relations between the invilved tools. But I still have problems to create a process for a multi module project, where all tests are separated from the implementation projects.
i.e. we have the following project structure:

RootProject
pom.xml

ProjectA
–> src
pom.xml

ProjectATest (depends on ProjectA)
–> test
pom.xml

ProjectB (depends on ProjectA)
–> src
pom.xml

ProjectBTest (depends on ProjectB)
–> test

pom.xml

ProjectITTests (depends on ProjectA and ProjectB)
–> test
pom.xml

Now I want to measure the code coverage with Jacoco. Everything I already tried didn't work. The code coverage report in the test projects doesn't contain the coverage of the sources that lie in different projects. i.e. the report of ProjectATest contains no coverage for sources of ProjectA.

Do you have any idea how to achive code coverage analysis with Jacoco and Maven?

REPLY

Petri

September 26, 2013, 20:50

I have to confess that I haven't used the JaCoCo Maven plugin in a multi module project like this. However, I did found a few websites which might be helpful to you:

- How can I get code coverage of an external java library with jacoco?
- JaCoCo in Maven Multi-Module Projects

REPLY

Benoît

November 27, 2013, 19:17

This is because your sources/classes are not available when JaCoCo is executed. I suppose that the JaCoCo plugin is executed on ProjectATest, ProjectBTest, and ProjectITTests ? Not on ProjectA, and ProjectB (which contains the sources/classes you want to analyse…) ?

Which kind of tests are performed on these projects ? UT or IT or both ?

REPLY

Olivier C.

December 4, 2013, 00:30

Really interesting article.
I have to use Jacoco for integration tests that call a Jetty server. I would like Jacoco returns code coverage of the classes used by Jetty. Here is my configuration :
https://groups.google.com/forum/#!topic/jacoco/K-PCEg5LLd0
Have you any experience / idea on how to configure the run of Jetty to make Jacoco returns the server side code coverage ?
Thanks.

REPLY

Petri

December 4, 2013, 20:21

I have no personal experience from this. However, I have one idea:

Have you tried to use the *run-forked* goal of the Jetty Maven plugin?

This starts Jetty in a new VM, and you can you can pass arguments to the forked VM by using the *<jvmArgs>* configuration element. You could try to pass the the value of the *failSafeArgLine* property to the forked VM by adding this line to the configuration section of the Jetty Maven plugin:

```
<jvmArgs>${failsafeArgLine}</jvmArgs>
```

Let me know if this solved your problem.

REPLY

Andrew Eells

December 4, 2013, 19:23

Excellent! Thanks very much Petri, had spent about a day messing around with our maven configuration to get the coverage reports split by Unit and Integration and also had problems including Groovy test results.
But all good with the help of your article.
@Stefan, I have a vanilla multi-module project that this is working with so will publish to Github if you still have an issue?
Thanks again @Petri, best regards, Andrew

REPLY

Petri Kainulainen

December 4, 2013, 20:06

Hi Andrew,

You are welcome! I am happy to hear that this blog post was useful to you.

Could you add the url of your vanilla multi-module project after you have published it? I would like to add the url to this blog post (and give kudos to you).

REPLY

umesh prajapati

December 13, 2013, 04:24

Hi petri,

I have been working on this for a week now and I am not sure why I am not getting any code coverage for acceptance test.

First: We have ant build project.
Second: We run our acceptance test using selenium

Steps I have tried to get code coverage:
Install jacocoant.jar and jacocoagent.jar in lib folder of our project.
In start-app.sh/start-reg.sh I have mentioned
javaagent=../pathto/jacocoagent.jar=destfile=../pathtostore/jacoco.exec,includes=com.abc.*

So, when I start my servers, it creates jacoco.exec . I execute my acceptance test, I thought after I execute my acceptanceTest, jacocoagent is suppose to generate code coverage. But, my jacoco.exec file is the same. I dont see any increasement on the file size even after the acceptanceTest is completed. I tried shutting down the servers after test is completed but still no change.

So, I am wondering how can I get code coverage for acceptanceTest. Did I miss anything?

REPLY

Petri

December 14, 2013, 01:28

I haven't used JaCoCo without Maven but I assume that the process has the same steps.

First, you have to generate the code coverage data by using the Java Agent.

When I run the the command *mvn clean test* (in the root directory of the example application) at command prompt, I see the following log line:

```
[INFO] surefireArgLine set to -
javaagent:/Users/loke/.m2/repository/org/jacoco/org.jacoco.agent/0
.6.3.201306030806/org.jacoco.agent-0.6.3.201306030806-
runtime.jar=destfile=/Users/loke/Projects/Java/Blog/maven-
examples/code-coverage-jacoco/target/coverage-reports/jacoco-
ut.exec
```

Is the *jacocoagent.jar* you mentioned the same jar which is used by my example application? Also, do you pass these arguments to the VM which is used to run your server?

Second, you have to create the actual code coverage report from the gathered code coverage data. You can create this reports by using Eclemma, Ant or Maven. It seems that there is no command line tool yet.

I hope that I could help you out although I couldn't provide the definitive answer.

REPLY

Petar

December 15, 2013, 02:29

Hi Petri,

are you aware of any way to measure the total coverage of both unit and integration tests? Some of my classes are really easy to unit test so I have unit tests for them, and some are quite hard so I've written integration tests. However with the setup you have provided it fails saying that class A has not enough unit test coverage (which is true cause it has integration tests and It's not that easy to write unit tests) or it fails saying class B has not enough integration test coverage (which is true – it has unit tests!!)..

Thanks for the nice article and keep up the good work!

REPLY

Petri

December 15, 2013, 11:41

Hi Petar,

You could try following these steps:

1. Put your integration tests to the *src/main/test* directory.
2. Remove the exclusion from the configuration of Maven Surefire plugin.
3. Remove the Maven Failsafe plugin from your POM file.

This should do the trick as long as your integration tests don't require that the application is deployed to a servlet container or an application server which is run in a separate VM.

REPLY

Tapio Rautonen

January 2, 2014, 00:21

Petri, nice and informative post how to use JaCoCo. If you are working with open source software, you should also check Coveralls, a code coverage hosting service.

I have created a maven plugin for the Coveralls service that can use either JaCoCo or Cobertura (and Saga soon) as the code coverage tool. Check the GitHub page of the Coveralls maven plugin and Coveralls docs for Java projects for more information.

REPLY

Petri

January 2, 2014, 10:12

Tapio, thanks!

Coveralls looks interesting, and I will take a closer look at it later today. The maven plugin looks good as well.

By the way, I hadn't heard about Saga before. Thank you for mentioning it. Maybe 2014 is the year when I start to write tests for my Javascript code.

REPLY

wiza

January 3, 2014, 13:58

Wow! Thank you very much for the really good description you are giving us.

As I'm a totally beginner, I am trying to setup Maven/Jacoco in order to publish the code coverage results to Sonar.
I have 1 question:
1) How do we set the JaCoCo execution data file? What is that?

Though, even following your post, I'm dealing with the same problem: maven-compiler doesn't find the 'sources' and I get an error: "No sources to compile" despite that maven-resources-plugin copies the right resources.

I get the following command line output:

```
$ mvn clean test
[INFO] Scanning for projects…
[INFO]
[INFO] ————————————————————————————————————————
```

```
[INFO] Building Code Coverage – Maven and JaCoCo running tests 1.0-SNAPSHOT
[INFO] ————————————————————————————————
[INFO]
[INFO] — maven-clean-plugin:2.5:clean (default-clean) @ example-java-maven —
[INFO] Deleting sonar_local/src/target
[INFO]
[INFO] — jacoco-maven-plugin:0.6.4.201312101107:prepare-agent (pre-unit-test) @
example-java-maven —
[INFO] surefireArgLine set to -javaagent:
.m2/repository/org/jacoco/org.jacoco.agent/0.6.4.201312101107/org.jacoco.agent-
0.6.4.201312101107-runtime.jar=destfile=sonar_local/src/target/coverage-
reports/jacoco-ut.exec
[INFO]
[INFO] — maven-resources-plugin:2.3:resources (default-resources) @ example-java-
maven —
[INFO] Using ‘UTF-8’ encoding to copy filtered resources.
[INFO] Copying 53 resources
[INFO] Copying 41 resources
[INFO]
[INFO] — maven-compiler-plugin:3.1:compile (default-compile) @ example-java-
maven —
[INFO] No sources to compile
[INFO]
[INFO] — build-helper-maven-plugin:1.8:add-test-source (add-integration-test-sources)
@ example-java-maven —
[INFO] Test Source directory:/sonar_local/src/test/tcl added.
[INFO]
[INFO] — maven-resources-plugin:2.3:testResources (default-testResources) @
example-java-maven —
[INFO] Using ‘UTF-8’ encoding to copy filtered resources.
[INFO] Copying 5 resources
[INFO]
[INFO] — maven-compiler-plugin:3.1:testCompile (default-testCompile) @ example-
java-maven —
[INFO] Nothing to compile – all classes are up to date
```

```
[INFO]
[INFO] — maven-surefire-plugin:2.16:test (default-test) @ example-java-maven —
[INFO]
[INFO] — jacoco-maven-plugin:0.6.4.201312101107:report (post-unit-test) @ example-
java-maven —
[INFO] Skipping JaCoCo execution due to missing execution data file
[INFO] ————————————————————————————————
[INFO] BUILD SUCCESS
[INFO] ————————————————————————————————
[INFO] Total time: 3.451s
[INFO] Finished at: Fri Jan 03 11:48:41 GMT 2014
[INFO] Final Memory: 12M/110M
[INFO] ————————————————————————————————
```

I would be really grateful if you could help me.
Thank you very much

hAppY NeW YeaR! !

REPLY

Petri

January 3, 2014, 20:46

Hi,

Thank you for your kind words! I really appreciate them.

The execution data file contains the code coverage information which is used to create the actual code coverage report. Its location is found from the configuration of the JaCoCo Maven plugin (unit tests and integration tests).

I have a few questions about your problem:

- Do you have any source code files in the *src/main/java* directory?
- What packaging type are you using in the `pom.xml` file (If you are using `pom` packaging type, read this StackOverflow question)?

REPLY

Justin

January 4, 2014, 18:38

Hi,

Thanks for the helpful post! One question: using this configuration (generating both unit and integration test Jacoco coverage reports) have you tried using it with the maven-site-plugin? I was only able to get either the unit OR the integration reports to display in the generated Maven site.

REPLY

Petri

January 4, 2014, 20:11

Hi,

I tried to use the maven-site-plugin when I wrote this blog post but I couldn't get it to generate the both reports. Then I found this pull request. It should fix this problem but at the time it was not merged yet.

It seems to be merged now. Maybe you could it out and see if it works? Here is a link to the commit which contains an example *pom.xml* file.

REPLY

Justin

January 7, 2014, 22:38

thanks for the reply Petri. i'll check it out

REPLY

Petri

January 7, 2014, 22:46

You are welcome!

I hope that it solves your problem. I should probably try it too because it would be a nice addition to the example application of this blog post.

REPLY

Manu

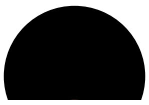
January 15, 2014, 23:08

Hi Petri,

I would like to generate the code coverage report on a server. For example, i have a jboss server application running in a QA environment and i'm running my tests from my laptop and the test is posting requests to the QA server. How would i get the code coverage on that server app?

Best,
Manu

REPLY



Petri

January 16, 2014, 19:57

I would do this by following these steps:

1. Attach Java agent to the used application server.
2. Create code coverage report from the execution coverage data recorded by the Java agent.

This should do the trick (in theory). I have never done this myself though.

REPLY

Manu

January 21, 2014, 23:22

I have one more question on the agent. If i run the jacoco agent on an app server for long time and generate the code coverage report without reset whenever i need it, would it likely to cause any out of memory error in app server? Does the tcp server option in jacoco agent save the metrics locally or does it keep all the information in memory which can cause out of memory error?

REPLY

Petri

January 22, 2014, 09:50

The JaCoCo documentation says that:

> The JaCoCo agent collects execution information and dumps it on request or when the JVM exits.

I have no idea about the inner workings of the JaCoCo agent but I would assume that it keeps the collected information in memory.

The documentation also says that if you run the agent in the file system mode, it writes the data to a file when JVM is terminated. Like you said, you can also run the agent in a mode where the data is written to a socket connection when it is requested (socket server or socket client modes) but I have no idea where the data is stored in this case. I would assume that it is stored in memory because the documentation doesn't say otherwise.

You could try to get a better answer to your question from the Google group called JaCoCo and EclEmma Users.

REPLY

kingsley

January 23, 2014, 07:12

Hi Petri,
I get a question and need your help. As I was trying to collect report，I encountered trouble in filtering class file which I'm not interested in. Do you have any suggestion on how this filtering process can be set through maven plugin？In your case，is it set to not execute unit test case that you are uninterested ?

REPLY

Petri

January 23, 2014, 09:31

Hi Kingsley,

You can exclude classes by configuring a list of class files which are excluded from the report.

Actually this week I had to exclude all JPA meta model classes from my unit test code coverage report (these are tested in integration tests). I did this by modifying the configuration of the execution which runs the *report* goal of the JaCoCo Maven plugin. My new configuration looks as follows:

```
<execution>
    <id>post-unit-test</id>
    <phase>test</phase>
    <goals>
        <goal>report</goal>
    </goals>
    <configuration>
        <dataFile>
            ${project.build.directory}/coverage-reports/jacoco-ut.exec
        </dataFile>
        <!-- Exclude the JPA meta model classes from the report -->
        <excludes>
            <exclude>**/*_.class</exclude>
        </excludes>
        <outputDirectory>
            ${project.reporting.outputDirectory}/jacoco-ut
        </outputDirectory>
    </configuration>
</execution>
```

I hope that this answered to your question.

REPLY

kingsley

January 24, 2014, 03:55

thanks for the reply ~

REPLY

google api console

February 1, 2014, 19:54

Normally I do not read post on blogs, however I would like to say that this write-up very compelled me to check out and do it! Your writing style has been amazed me. Thank you, quite great post.

REPLY

Petri

February 1, 2014, 21:00

Thank you for your kind words. I really appreciate them!

REPLY

Greg

February 13, 2014, 02:59

Hi Petri,

Question: How to get code coverage using the jacoco coverage engine while executing the robot acceptance test cases? Usually robot test cases are run after the UT and IT. How do I integrate the jacoco in such case. Really appreciate for any input.

Thanks
Greg

REPLY

Petri

February 13, 2014, 21:52

Hi Greg,

how do you run the robot test cases? Are you using the Robot Framework Maven plugin? I think that you have to

1. Ensure that the JaCoCo runtime agent is running and gathering execution data when the robot test cases are executed.
2. Create the code coverage report after the robot test cases haven been executed.

Unfortunately this was the easy the part. The hard part is that I have no idea how you can accomplish this.

I took a quick look at the documentation of the plugin's run goal and I couldn't find a way to pass custom arguments to the plugin (maybe the `argumentFile` configuration element is used for this purpose?) If it is, you could try to add the following line to that file (change the directory paths):

```
-javaagent:/Users/loke/.m2/repository/org/jacoco/org.jacoco.agent/0.6.3.201306030806/org.jacoco.agent-0.6.3.201306030806-runtime.jar=destfile=/Users/loke/Projects/Java/Blog/maven-examples/code-coverage-jacoco/target/coverage-reports/jacoco-ut.exec
```

This starts the JaCoCo runtime agent and configures the location of the file which contains the execution data.

The second step is more problematic. You have to trigger the report goal of the JaCoCo Maven plugin after your robot test cases have been executed. Maybe you could configure the JaCoCo Maven plugin to invoke this goal when you create site for your project or something like this? I know it isn't the optimal way to do this but at the moment I cannot figure out a better solution. :(

REPLY

Spencer

February 27, 2014, 21:40

Thanks for this — I was able to quickly add JaCoCo to my POM. Unfortunately, JaCoCo is incompatible with PowerMock's byte code rewriting (https://github.com/jacoco/eclemma/issues/15). Luckily, Cobertura Maven plugin 2.6 works with Java 7.

REPLY

Raj

March 27, 2014, 07:11

Thanks for this , however, when I ran mvn clean verify -P integration-test or mvn clean verify -P all-tests, the jacoco-it folder has index.html but shows 0% test coverage.
My project code is under – src/main/java and my tests are under src/test/java
can you please help?

REPLY

Petri

March 27, 2014, 09:58

The tests should be placed as follows:

- Integration tests must be placed to the *src/integration-test/java* directory.

- Unit tests must be placed to the *src/test/java* directory.

The test coverage is zero percent because you try to generate a code coverage report for integration tests but your tests are in the "wrong" directory.

You can solve this problem in two ways:

- Move the tests to the *src/integration-test/java* directory. You should do this only if the tests are integration tests and not unit tests.
- Create the code coverage report by using the command *mvn clean test*. You should this only if the tests are unit tests.

REPLY

Raj

March 27, 2014, 18:18

I copied all my integrations tests to src/integration-test/java directory but that too didnt help.
I dont need code coverage report for the unit tests because we dont have any.
Tests run: 13, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] — jacoco-maven-plugin:0.6.3.201306030806:report (post-integration-test) @ #####
[INFO] — maven-failsafe-plugin:2.16:verify (integration-tests) @ ######
[INFO] ————————————————————————
[INFO] BUILD SUCCESS
[INFO] ————————————————————————

the report still shows as 0% code coverage for my integration tests.

Please help!

REPLY

Petri

March 27, 2014, 20:26

Does the tested code run in the same JVM than the test, or are you testing an application which is deployed to a server?

If the code is deployed to an application server, I would follow the steps given in this answer (I haven't tried this myself so I have no idea if it works).

If the tested code runs in the same JVM than the test, I probably need to your *pom.xml* so that I can figure out what is wrong. You can paste it to pastebin.com and add a link to your comment (don't paste it here because Wordpress eats XML tags and the remaining text is pretty much useless).

REPLY

Raj

March 31, 2014, 06:39

am not sure about the same JVM question that you asked..
our application code and our integration tests are under the same folder structure…so when I run the mvn clean verify -P integration-test command on the terminal, the tests run against the code/service running on localhost .
the tests run good, but the report shows 0% coverage… :-(

Raj

March 31, 2014, 10:25

this is what I do:

start my service locally, run my integration tests using mvn clean verify -P integration-test and then look into the target/jacoco-it folder for the report
folder structure:

I want to view only the integration test results , so i have used your stuff under the— Configuring The Code Coverage Report for Integration Tests
can you please assist further?

Petri

March 31, 2014, 12:40

Hi Raj,

If you start the tested service manually, it is probably running in a different JVM than the integration tests (and the JaCoCo agent). That is why the code coverage is zero percent.

You can fix this by starting the JaCoCo agent in the JVM which runs the tested code. If you use an application server, you can follow the instructions given in this comment (I haven't done this myself I have no idea if this really works).

Bhupesh Bansal

April 6, 2014, 03:40

Thanks !! I was looking to add code coverage for integration tests. Jacoco and your blog was a perfect match !!

REPLY

Petri

April 6, 2014, 11:08

You are welcome! I am happy to hear that this blog post was useful to you.

REPLY

atul gupta

May 13, 2014, 09:14

thanks a lot. it was a wonderful article and did help me.

I learnt a lot. thanks for all your efforts.

Cheers

Atul

REPLY

Petri

May 13, 2014, 18:01

You are welcome. I am happy to hear that this blog post was useful to you.

REPLY

Rajesh

May 30, 2014, 13:35

Petri,

Thanks for your blog. I was looking for Integration test code coverage and your blog is on spot on what I was looking for.

However just out of interest, are you aware of any tool which does code coverage at run time? For example, when we run automated service test packs with Soap UI tool or such tools against some web service, is it possible to find out the number of lines, methods or business blocks executed at service during run time? If we have anything of this sort, It would help to see if we have enough service test packs to cover all scenarios and also it would help to get rid of redundant code which is never used.

Any idea on this?

Thanks
Rajesh

REPLY

Petri

June 2, 2014, 20:51

Hi Rajesh,

I haven't tried this myself but it should be possible (at least in theory).

By the way, thank you for posting this comment. I have never thought about using JaCoCo in this way, and this is actually a pretty good way to identify unused code which can be deleted.

REPLY

Heath Borders

June 12, 2014, 00:13

Thanks a million.
This is much better documentation than jacoco's official docs.

REPLY

Petri

June 12, 2014, 22:53

Thank you! I am happy to hear that this blog post was useful to you.

REPLY

Tomas

July 8, 2014, 10:58

Hi Petri,

thank you very much for this useful post!

I have an issue about excluding classes from JaCoCo reports. I copy-pasted you configuration for
unit test coverage (don't have any ITs) only extra thing I added to configuration of both executions is

MersenneTwisterFast

The exclude for jacoco:coverage works fine but the MersenneTwisterFast class is still shown in reports with 0% coverage. Any idea why it does not work? I searched the web and only thing I found was people missing out the exclude for reports, but I do have it there.

REPLY

Tomas

July 8, 2014, 11:05

Solved,

jacoco:coverage

**/MersenneTwisterFast.java

jacoco:report

**/MersenneTwisterFast.class

Sorry for bothering you.
Keep up the good work!

REPLY

Petri

July 8, 2014, 15:19

Tomas,

It is good to hear that you were able to solve your problem! And no worries, feel free to ask questions if you have problems or if you cannot understand something. Unfortunately I couldn't help you this time since I was outside chopping firewood.

REPLY

Maximus

July 18, 2014, 16:57

Hi Petri

Excellent article. Your way of writing is very concise and clear. I have one question.

I am trying to build a framework for my project. We use restful web services with json. Now this is what I want to do.

1. Create a test database. There I will store testcase id, target url, json request, expected response etc. This can be more elaborate and exhaustive if required.
2. When I start to run my test suite, there will be a component which will read all test cases from database.
3. It will fire each test case and collect actual response and will update a table with actual response.
4. Finally there will be another component which will read this table and prepare an html report indicating total test cases run, failed and passed test cases.

What I would like to know is, are there any issues with this approach?
Second, is there any way if, after building this framework, I can get coverage percentage along with failed and passed test cases? Will the plugin which you described above help me get the coverage percentage?

The scenario I am describing would happen on QA environment with servers running.
Appreciate if you would let me know your thoughts.

Thanks for this wonderful article again.
Cheers

REPLY

Petri

July 18, 2014, 18:55

> What I would like to know is, are there any issues with this approach?

Well, it seems that you have to write a lot of plumbing code to get this done, but this might be a reasonable approach if you want to create new tests without writing any code. I have never done anything similar myself so unfortunately I cannot give any specific advice. Are you planning to extend an existing framework (such as Cucumber or Robot), or do you want write your own framework?

> Is there any way if, after building this framework, I can get coverage percentage along with failed and passed test cases?

To be honest, I don't know the answer to this question because I don't know if JaCoCo can differentiate the failed and passed test cases. I assume that it cannot do this because it cannot know whether your test case fails or passes.

> Will the plugin which you described above help me get the coverage percentage?

It can help you to get the code coverage percentage but since the JaCoCo agent must be run in the server VM, you would have to download the execution data files from the server and figure out a way to generate the coverage report after the files have been downloaded. I think that this is doable but since I have never done this myself, I cannot give you any specific advice.

REPLY

KSM

July 26, 2014, 08:36

Hi Petri,

Thanks for the awesome write-up. Basically, I am working on a program/tool which would invoke use JaCoCo on a package to generate reports and use those generated reports to get selected fields from it and make a whole new report. I know this sounds redundant but I have to do it anyways. I haven't yet started working on it yet as I am thinking of quick and efficient ways of doing this. Since I am a complete beginner to this area, I wanted some ideas/suggestions from you. Can you help me out with this? Thanks.

REPLY

Petri

July 26, 2014, 11:39

There are a few things that come to my mind:

- I assume that your tool would create those reports from the execution metadata gathered by the JaCoCo agent? The reason why I ask this is that the JaCoCo agent must be running in the same VM than the "inspected" code.
- Do you think that you could use the the *report* goal of the JaCoCo Maven plugin when you create the first report? This way you wouldn't have to read the execution metadata yourself (I tried to find its specification from the Google but I couldn't find anything). After the first report is created, you can find the information you need and create the another report.

I know that this isn't much, but I hope that it helps you to get started.

REPLY

Slim

July 28, 2014, 18:09

Thanks for this wonderful article

JaCoCo not consider spring data interface repository to calculate coverage metrics
How tell jacoco to cover Spring data repository integration test ?

Thanks

REPLY

Petri

July 29, 2014, 15:54

The JaCoCo FAQ states that:

> Java interface methods do not contain code, therefore code coverage cannot be evaluated. Indeed code coverage is recorded for the implementation classes. The same applies to abstract methods in abstract classes.

It seems that there is no way to include repository interfaces in your code coverage report :(

REPLY

Slim

July 30, 2014, 15:47

Thanks
For now I added in ineterfaces constants that contains test values for the input parameters.
This is not propore!

REPLY

Shankar

August 20, 2014, 19:56

Hi Petri,
Excellent article.
I am trying to get the integration coverage report using jacoco for my selenium web test cases which is a seperate testcases module(using failsafe plugin). Application is deployed in remote tomcat server. I am executing the test cases from another server.
How can i get the IT coverage using maven plugins.

Thanks

Shankar

REPLY

Petri

August 20, 2014, 20:19

Hi,

I proposed one possible solution to your problem in this comment. Unfortunately I haven't done this myself so I am not sure how you can get the execution data from the remote server before you create the code coverage report.

REPLY

Shankar

August 20, 2014, 20:46

Hi,
Thanks,
I have added the java agent to my tomcat JAVA_OPTS where my application is installed
Once i start the tomcat , jacoco.exec is created. Also i have the classdumpdir to the java_opts.
After doing some operation , classdumpdir has dumped the classes but the jacoco.exec has nothing.
So my coverage report is Zero

Thanks

REPLY

Petri

August 23, 2014, 10:55

It seems that I should investigate this problem and write a blog post about it. I added this blog post idea to my Trello board, and I will write this blog post after I have finished a few other blog posts that I need to write before this one.

REPLY

Divya Deepak

August 26, 2014, 13:59

Hi Petri,
Can you give me some guideline in integrating cobertura and Fitnesse through Maven?

REPLY

Petri

August 26, 2014, 18:44

I have never used FitNesse myself but I think that your best chance is to use the fitnesse-launcher-maven-plugin. There is another FitNesse Maven plugin out there but it seems that is not being developed anymore.

When I was still using Cobertura, I integrated it to my build by using Cobertura Maven plugin. Its documentation has a page that describes how you can use it. You might also want to check out a blog post titled: Maven + Cobertura code coverage example.

REPLY

Divya Deepak

August 27, 2014, 08:48

Thanks for the prompt reply! I will go through the link that you provided.

REPLY

Marco

November 17, 2014, 15:58

Hi Petry

really nice article and it helped me a lot.

I still have a question about tests with spring-test library and the automatic Spring context. I'm having an issue because in my test class I'm using spring-test with the annotation for the configuration as

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration()
```

Using your approach with the added integration-test dir under src and the build-helper-maven-plugin works fine until the test class tries to load the myTest-context.xml. then it throws an exception saying that there's not any -context.xml file. I tried to add in the build-helper plugin a resources directory where I put the *-context.xml file but it doesn't work.

Do you have some idea of how I can make it ?

Thanks in advance
Marco

REPLY

Petri

November 17, 2014, 20:50

Hi Marco,

Have you read my blog post titled: Integration Testing with Maven? It describes how you can create a Maven build script for a project that uses separate directories for unit and integration tests. Also, if you want to create a separate application context configuration file for your integration tests, you can add this file to a separate resource directory and add the resources directory to the classpath.

I tried to look for an example that would use this approach but I couldn't find anything. In any case, you should be able to create a separate application context configuration file for your integration tests by following these steps:

1. Create the application context configuration file (*myTest-context.xml*) and put it to the *src/main/integration-test/resources* directory.
2. Add the *src/main/integration-test/resources* directory to the classpath.
3. Configure your integration test to use your application context configuration file. In other words, annotate your test class with the following annotation: `@ContextConfiguration(locations = {"classpath:myTest-context.xml"})`.

I hope that this answered to your question.

REPLY

Marco

November 17, 2014, 22:34

Hi Petri

thanks a lot for your quick reply and yes you answered my question.

Indeed I saw your blog post but I think I missed that part. :-(I've just implemented now and it fixed it. So here you can find the example on my github project http://github.com/marcomaccio/cxfatwork. It's implemented in the branch jacoco-int-tests. I'll make it better in these coming days.

I've mentioned you in the commit if you don't mind. (but I don't know if it worked)

It's a complete example of REST API and I trying to test it with your approach with some integration test and some cucumber test for user acceptance tests.

I'll try to add the DAL in the next week so it would a be a workable example for CRUD API

Thanks a lot again.
Marco

REPLY

Petri

November 17, 2014, 22:54

Hi Marco,

You are welcome. It is good to hear that you were able to solve your problem.

By the way, do you think that I should update my blog post (Integration Testing with Maven)? I think that it might be a good idea because you couldn't find the information that was relevant for you.

REPLY

Marco

November 17, 2014, 23:33

Hi Petri

I think it will complete your nice explanation. It’s a bit tricky point personally I didn’t get that I needed a second execution but it could be enough add somehow the resources dir in the first execution that you said in the post.

That part is not well explained in the plugin documentation so indeed it will help and it will straightforward.

Funny to read in your comment that another Marco asked you something similar ;-)

Have a nice eve
Marco

Petri

November 18, 2014, 18:34

Hi Marco,

Thank you for the feedback. I will update that blog post when I have got the time to do it.

Zala

November 21, 2014, 04:37

Hi! You saved my night, man! Thanks a lot!

REPLY

Petri

November 23, 2014, 23:59

You are welcome! I am happy to hear that this blog post was useful to you.

REPLY

Zeia

December 22, 2014, 10:48

Hello Petri,

Nice writeup on the topic indeed.

Can you please tell me if this solutions works with coverage reports for Selenium tests ?

I need to get the JaCoCo generated report under the "Integeration Reports" section in Sonar dashboard.

Best Regards,

Zeia.

REPLY

Petri

December 23, 2014, 22:00

Hi,

Thank your for your kind words. I really appreciate them.

> Can you please tell me if this solutions works with coverage reports for Selenium tests ? I need to get the JaCoCo generated report under the "Integeration Reports" section in Sonar dashboard.

I have never done this myself, but maybe these links will help you to solve your problem:

- JaCoCo Selenium test code coverage and import to Sonar using Ant
- Test Automation with Selenium WebDriver and Selenium Grid – part 3: Continuous Integration
- Measure Code Coverage by Integration Tests with Sonar
- JaCoCo with Sonar

REPLY

Sanjay Singh

January 6, 2015, 03:49

Hey Petri,
Not sure if my previous comment made it through, just wanted to ask you if you know how to get the jacoco report working for an ejb3 project?

Thanks

REPLY

Petri

January 7, 2015, 17:55

I have not done this myself, but it seems that the Arquillian project has an extension that adds JaCoCo support.

You should read the following articles:

- Arquillian Guides – Getting Started
- Arquillian Guides – Getting Started: Rinse and Repeat
- Code Coverage Arquillian and Jacoco

I hope that this answered to your question.

REPLY

omar

April 8, 2015, 04:45

Thank you for that sir but, it will be more Good to add any movie explain that in details Thank you again

REPLY

Petri

April 8, 2015, 09:51

You are welcome!

Also, thank you for the feedback regarding the format of my content. I managed to find this screencast: JaCoCo in Maven Multi-Module Projects. Maybe it will help you to understand this topic.

REPLY

shanthisagar

April 14, 2015, 13:27

Hi Petri,

Thanks for the wonderful explanation.
Currently I could observe that the analysis is being done for all the java classes included in the project for example in src/main/java, target/generated-sources, src/generated/java.
Is there any way that we can ask jacoco plugin to generate reports for java files only in src/main/java.

REPLY

Petri

April 14, 2015, 20:38

Hi Shanthisagar,

Check out this Stack Overflow answer. It describes how you can exclude classes from the generated report.

REPLY

shanthisagar

April 15, 2015, 12:29

Hi Petri,

Thanks for the link.
But I was more of wondering if we could include or exclude as per the src/main/java package.

REPLY

Petri

April 15, 2015, 13:28

You can either include or exclude class files (or both). The only difference between these scenarios is the name of the XML element.

If you want include class files, you must use the `includes` and `include` elements.

If you want to exclude class files, you must use the `excludes` and `exclude` elements.

The rules for creating the actual values that are included or excluded are described in this Stack Overflow answer.

REPLY

shanthisagar

April 16, 2015, 02:37

Thanks Petri for that.
But is there any way to include and exclude as per the source files not the class files?

Petri

April 16, 2015, 17:50

The JaCoCo documentation states that you have to specify a list of class files that are either excluded from the report or included in the report. That is why I assume that this is not possible.

Little Santi

April 29, 2015, 19:39

Thanks a lot for the explanation!

REPLY

Petri

April 29, 2015, 19:51

You are welcome!

REPLY

leela

May 6, 2015, 20:56

Configured jacoco for unit test coverage as described above.But its showing coverage report for only testng unit test case and ignoring powermock unit test cases. Is there a way to include powermock unit test cases as part of coverage?

REPLY

Petri

May 6, 2015, 22:40

I haven't tried this out, but this StackOverflow answer might solve your problem. You might want to also check out the following bug reports:

- PowerMock disables EclEmma code coverage #15
- Coverage of runtime modified classes #51

REPLY

GoEatLearn

June 9, 2015, 07:29

Sorry for earlier reply post – it wasn't reply but a comment:

Made a pictorial explanation for same JaCoCo usage with sonar:

http://www.goeatlearn.com/2015/06/code-coverage-using-jacoco-and-sonar.html

Thank you.

REPLY

Petri

June 9, 2015, 18:23

Thank you for sharing your blog post. I am sure that it is useful to someone.

REPLY

madhu

July 8, 2015, 09:50

Dear Petri,
i have configured the POM.xml of web application & run the in command promft following commands,
but i didn't the any report folder in target/site/jacoco-ut
please let me know the how will get the unit test report as well IT report
Thanks in advance
Madhu
my pom xml is :

Update: I removed the pom.xml because Wordpress removed all XML tags from it –

Petri

REPLY

Petri

July 8, 2015, 10:53

Unfortunately Wordpress removed all XML tags from your comment. This means that it was impossible to figure out if the problem is caused by *pom.xml* file.

Anyway, I have couple of questions for you:

- Which command did you run at comment prompt?
- Are you using the same Maven profiles than the example?
- Which Java version are you using? I noticed that this example didn't work with Java 8, but I fixed that by updating the JaCoCo Maven plugin to version 0.7.5.201505241946.

REPLY

Gaurav

November 8, 2015, 08:59

Hi Petri,
Thanks for sharing this, it helped a lot, more or less I followed your tutorials to generate integration tests, but jacoco is not generating report for integration tests however it generates the report for unit tests. The difference only in my configuration is the integration tests package structure,
When I run command "$ mvn clean verify -P integration-test ", its running all integration tests but at the end report are also getting generated but it shows zero percentage line and branch coverage , could you please help me,

Attached screenshot of report

http://postimg.org/image/63xl8vhx9/

Below is my pom.xml

org.apache.tomcat.maven

tomcat7-maven-plugin

2.2

http://localhost:9080

localhost

/joyious

9080

tomcat7-run

run-war-only

pre-integration-test

true

tomcat7-shutdown

shutdown

post-integration-test

Best Regards

Gaurav

Update: *I removed the irrelevant parts of the pom.xml file – Petri*

REPLY

Petri

November 8, 2015, 10:54

Hi Gaurav,

Are your integration tests run against a code that is deployed to a Tomcat servlet container? If so, this approach won't work because the JaCoCo agent cannot gather the execution coverage data.

However, I noticed that the JaCoCo Maven plugin has a *prepare-agent-integration* and *report-integration* goals that might be useful to you.

REPLY

Abhijit

December 7, 2015, 12:54

Hello Petri,

i am trying to include jacoco for my web application which is based on Maven build, jboss server start up. My requirement is to generate code coverage report after the post build i.e. once application is builded ear/war file will generate. this file will be deployed to jboss server. then i need to navigate to GUI / application in local enviroment to launch test cases. then after that based on test cases performed coverage report should generrate.

Could you please guide me how should i proceed ?

REPLY

Petri

December 7, 2015, 19:47

Hi,

I have never done this myself, but I assume that you can pass the JaCoCo agent as a VM option when you start the JBoss server. Take a look at these StackOverflow questions:

- Code coverage of JBoss AS 7 testsuite, using JaCoCo – no data in jacoco.exec files

- How to attach JaCoCo Agent to application server

REPLY

Stephane

December 16, 2015, 15:03

One again I got a helpful hand av Petri to whom I say Than You !!

I could quite easily add JaCoCo to my multi module Maven project.

I still need to see if the integration tests give their output, but the unit tests report has already helped me.

REPLY

Petri

December 16, 2015, 21:57

You are welcome. Let me know if you cannot get the code coverage results of your integration tests.

REPLY

Stephane

December 17, 2015, 21:55

Hi Petri,

I do have a nice report for the integration test code coverage, but there is a 0 percent coverage for each and every source file. And I do have integration tests, and these are run successfully.

When I click on the "Sessions" link in the top right corner of the HTML report I cannot see my classes in the list.

Update: I removed the XML from this comment as requested – Petri

REPLY

Stephane

December 17, 2015, 21:59

Hi Petri,

Sorry for the miserable above post. I see the markup is not allowed. Feel free to edit or delete it.

I have posted a question on SO as it allowed me to post the markup.

http://stackoverflow.com/questions/34343214/zero-percent-coverage-by-integration-tests

Cheers,

REPLY

Petri

December 17, 2015, 22:53

Hi Stephane,

What kind of integration tests do you have? Are they run against an application that is deployed into a servlet container? The reason why I ask this is that the agent must be attached to the VM that runs the tested code. If the code coverage is zero percent, the agent might be attached to the wrong VM.

REPLY

Stephane

December 18, 2015, 11:49

Hi Petri,

The tests are run by the jetty plugin. I have updated my SO question with the Maven Jetty plugin configuration.

From your comment I understand the Jetty plugin runs with its own VM and… the JaCoCo could be running with another one ?

REPLY

Petri

December 18, 2015, 21:02

Hi Stephane,

> From your comment I understand the Jetty plugin runs with its own VM and… the JaCoCo could be running with another one ?

This depends from the goal of the Jetty Maven Plugin that is invoked before you run your integration tests. I noticed that you invoke both the `start` and the `run-forked` goals of the Jetty Maven plugin before your integration tests are run. Do you really need them both? If you use only the `run-forked` goal, you could try attaching the agent by using JVM arguments (I took a look at your configuration and noticed that you already did this).

If this approach doesn't work, you could take a look at the `prepare-agent-integration` goal of the JaCoCo Maven plugin. I have never used it myself, but it looks like the right choice for this scenario.

Stephane

December 21, 2015, 15:20

Hi Petri,
I had been using these goals already in fact. I can start the Jetty server in fork mode all right, but the percentage results are all at zero again http://stackoverflow.com/questions/34343214/how-to-have-the-maven-build-wait-for-the-jetty-server-to-start-in-forked-mode-be

Petri

December 23, 2015, 22:57

Hi Stephane,

I have to confess that I am out of ideas. However, since this seem to be a somewhat popular scenario, I will try to create an example project that uses this approach. If I am successful, I will write a new blog post about it.

Leave a Comment

Name (required)
Email (not required)
Website (not required)

SUBMIT

PREVIOUS POST: WHAT I LEARNED THIS WEEK (WEEK 32/2013)
NEXT POST: WHAT I LEARNED THIS WEEK (WEEK 33/2013)

GET FREE EBOOK

Subscribe my email newsletter **AND** you will get my eBook: Writing Integration Tests for Spring Powered Repositories **FOR FREE**.

Email Address...

SUBSCRIBE

I will never sell, rent, or share your email address.

LEARN NEW SKILLS

Getting Started With Gradle

Maven Tutorial

Spring Data JPA Tutorial

Spring Data Solr Tutorial

Spring From the Trenches

Spring MVC Test Tutorial

Spring Social Tutorial

Using jOOQ with Spring

Writing Clean Tests

Writing Tests for Data Access Code

SEARCH

enter search term and press enter

FROM THE BLOG

Recent Popular Favorites

Spring Batch Tutorial: Reading Information From a File

Java Testing Weekly 6 / 2016

Spring Batch Tutorial: Getting the Required Dependencies With Gradle

The Best Comments of January 2016

Java Testing Weekly 5 / 2016

CATEGORIES

- Programming (136)
  - Apache Hadoop (3)
  - Apache Wicket (4)
  - Asciidoctor (1)
  - Clean Code (1)
  - Gradle (9)
  - jOOQ (4)
  - Maven (8)
  - Solr (9)

- Spring Framework (67)
- Testing (21)
- Tips and Tricks (16)
- Unit Testing (5)

Reviews (3)

Software Development (66)

- Design (10)
- General (8)
- Learning (21)
- Processes (21)
- Technology Evaluation (6)

Uncategorized (17)

Weekly (6)

Writing (2)