

Tutorial – Coveralls

Inhaltsverzeichnis

Einführung	1
Voraussetzungen	1
Nutzung von Coveralls.....	2
Inkludieren des Coveralls-Maven-Plugins	2
Verwendung von Coveralls mittels Jacoco	3
Jacoco	3
Inkludieren des Jacoco-Plugins.....	3
Anpassen der Travis-Konfiguration	3
Exkludieren von Code	3
Verwenden von coveralls.io mittels Cobertura.....	4
Cobertura	4
Inkludieren des Cobertura-Plugins.....	4
Anpassen der Travis-Konfiguration	4
Exkludieren von Code	5
Verwenden von coveralls.io mittels Sage	5
Saga	5
Inkludieren des Sage-Plugins.....	5
Anpassen der Travis-Konfiguration	6
Exkludieren von Code.....	6
Kombination mit anderen Code-Coverage-Tools.....	6
Weiterführende Informationen	7

Einführung

Dieses Tutorial beschreibt die Verwendung des online Code-Coverage-Tools coveralls.io zur Widerspiegelung und Nachverfolgung der Testabdeckung bei Softwareprojekten. Es wird beispielhaft vorgeführt welche Schritte notwendig sind, um ein Projekt in der Programmiersprache Java unter Verwendung des Build-Management-Tools Maven und dem Deployment-Tool Travis CI an die Plattform coveralls.io anzubinden.

Voraussetzungen

Voraussetzung für dieses Tutorial ist, dass es sich bei eurem Projekt um ein Java Projekt handelt, welches das Build-Management-Tool Apache Maven sowie Travis CI als Deployment-Tool nutzt.

Des Weiteren solltet ihr euch schon bei coveralls.io registriert haben und dort das Projekt zu eurem Account hinzugefügt haben.

Nutzung von Coveralls

In den folgenden Abschnitten wird euch gezeigt wie und an welchen Stellen ihr euer Java Projekt anpassen müsst, um coveralls.io für euer Projekt zu verwenden. Dazu ist es zum einen notwendig ein entsprechendes Maven-Plugin in eure pom.xml einzubinden, welches die Anbindung an coveralls.io ermöglicht.

Des Weiteren müsst ihr eines der drei hier vorgestellten Code-Coverage-Tools (Jacoco, Cobertura, Saga) via Maven-Plugin in euer Projekt integrieren, da diese die eigentlichen Reports zur Testabdeckung erstellen die coveralls.io dann verwendet.

Zuletzt müsst ihr noch eure Travis-Konfigurationsdatei .travis.yml anpassen, so dass nach jedem Deployment-Prozess von Travis Reports, bezüglich der Testabdeckung erstellt und an coveralls.io übermittelt werden. Dadurch wird für euer Projekt bei jedem Deployment-Prozess die entsprechende Testabdeckung automatisch erfasst und durch coveralls.io festgehalten.

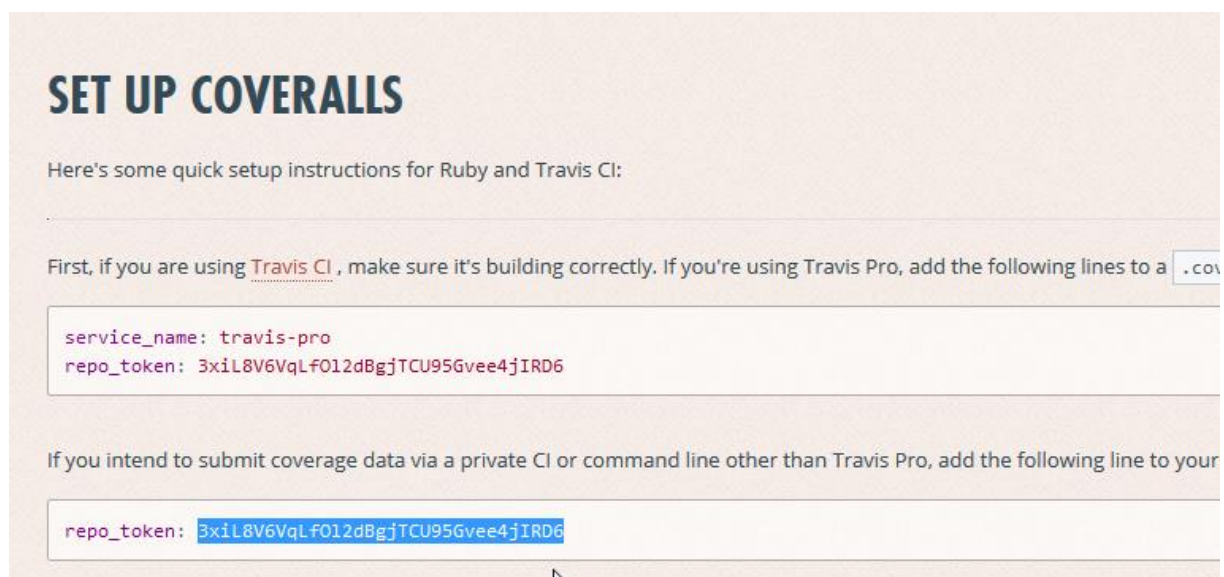
Inkludieren des Coveralls-Maven-Plugins

Unabhängig davon welches Code-Coverage-Tool ihre letztendlich benutzt, Voraussetzung für die Anbindung an coveralls.io ist, dass ihr das Coveralls-Maven-Plugin in eure pom.xml hinzufügt.

Dazu müsst ihr folgenden Abschnitt in eure pom.xml im Teil <plugins> ... </plugins> eintragen:

```
30
31     <plugin>
32       <groupId>org.eluder.coveralls</groupId>
33       <artifactId>coveralls-maven-plugin</artifactId>
34       <version>4.1.0</version>
35       <configuration>
36         <repoToken>3xiL8V6VqLf012dBgjTCU95Gvee4jIRD6</repoToken>
37       </configuration>
38     </plugin>
39
```

Wichtig hierbei ist, dass ihr euer projektspezifisches Repository-Token im Tag <repoToken> ... </repoToken> eintragt, da das Plugin sonst euer Projekt innerhalb von coveralls.io nicht findet. Das Repository-Token erhaltet ihr wenn ihr euer Projekt innerhalb von coveralls.io zu euren Projekten/Repositories hinzufügt. Bei diesem Vorgang sollte euch in etwa das folgende angezeigt werden:



SET UP COVERALLS

Here's some quick setup instructions for Ruby and Travis CI:

First, if you are using **Travis CI**, make sure it's building correctly. If you're using Travis Pro, add the following lines to a `.coveralls.yml` file:

```
service_name: travis-pro
repo_token: 3xiL8V6VqLf012dBgjTCU95Gvee4jIRD6
```

If you intend to submit coverage data via a private CI or command line other than Travis Pro, add the following line to your `.coveralls.yml` file:

```
repo_token: 3xiL8V6VqLf012dBgjTCU95Gvee4jIRD6
```

Entnehmt das euch angezeigte Repository-Token und tragt es in eure pom.xml ein.

Verwendung von Coveralls mittels Jacoco

Dieser Abschnitt beschreibt welche Anpassungen notwendig sind um coveralls.io auf Basis des Code-Coverage-Tools Jacoco zu verwenden.

Jacoco

Jacoco ist ein Code-Coverage-Tool zur Ermittlung der Testabdeckung von Java-Dateien. Jacoco ist ein sehr „strenges“ Tool und erfordert das jede Zeile Code durch entsprechende Tests abgedeckt ist. Dies beinhaltet auch „trivialen“ Code, wie einfache Zuweisungen, Getter & Setter, etc. Die einzige Möglichkeit Code von der Überprüfung auszuschließen, ist das Exkludieren gesamter Quelldateien. Dafür ist Jacoco aktuell das einzige Tool welches Java 8 unterstützt. Des Weiteren arbeitet Jacoco ausschließlich auf dem Bytecode (class-Dateien), so dass die Source-Dateien nicht zwingend benötigt werden um die Testabdeckung zu ermitteln.

Inkludieren des Jacoco-Plugins

Zur Verwendung von Jacoco gibt es ein entsprechendes Maven-Plugin welches durch folgenden Eintrag in die pom.xml zum Projekt hinzugefügt wird:

```
38
39
40     <plugin>
41       <groupId>org.jacoco</groupId>
42       <artifactId>jacoco-maven-plugin</artifactId>
43       <version>0.7.6.201602180812</version>
44       <executions>
45         <execution>
46           <id>prepare-agent</id>
47           <goals>
48             <goal>prepare-agent</goal>
49           </goals>
50         </execution>
51       </executions>
52     </plugin>
53   </plugins>
```

Anpassen der Travis-Konfiguration

Damit coveralls.io die von Jacoco erstellten Code-Coverage-Reports beim Deployment-Prozess von Travis auch übermittelt werden, muss in die Travis Konfiguration .travis.yml folgender Eintrag hinzugefügt werden:

```
13
14   after_success:
15     - mvn clean test jacoco:report coveralls:report
```

Durch diesen Eintrag werden nach dem erfolgreichen Deployment die Code-Coverage-Reports durch Jacoco erstellt und durch das Coveralls-Maven-Plugin an coveralls.io gesendet.

Exkludieren von Code

Wie bereits erwähnt ist es bei Jacoco lediglich möglich komplette Dateien von der Überprüfung auszuschließen, dies geschieht mittels folgender Einträge in die pom.xml:

```

41 <version>0.7.8.201602180812</version>
42 <configuration>
43   <excludes>
44     <exclude>**/Main.class</exclude>
45   </excludes>
46 </configuration>
47 <executions>

```

Mit diesem Eintrag wird die Klasse „Main“ repräsentiert durch die Datei „Main.class“ von der Erfassung der Testabdeckung ausgeschlossen.

Verwenden von coveralls.io mittels Cobertura

Der folgende Abschnitt beschreibt welche Anpassungen am Projekt vorgenommen werden müssen um coveralls.io auf mit dem Code-Coverage-Tools Cobertura zu verwenden.

Cobertura

Bei Cobertura handelt es sich wie auch bei Jacoco um ein Code-Coverage-Tool zur Ermittlung der Testabdeckung von Java Dateien. Im Unterschied zu Jacoco ist Cobertura nicht ganz so „streng“ und bietet mehr Möglichkeiten einzelne Codeteile von der Überprüfung auszuschließen. So ist es möglich eine Einstellung zu setzen, so dass „trivialer“ Code von der Überprüfung ausgeschlossen wird. Zudem verfügt Cobertura über eine Annotation mit der einzelne Code-Abschnitte wie z.B. Methoden von der Überprüfung ausgeschlossen werden. Dafür bietet Cobertura leider keine Java 8 Unterstützung.

Inkludieren des Cobertura-Plugins

Zur Verwendung von Cobertura gibt es ähnlich wie für Jacoco ein Maven-Plugin welches durch folgenden Eintrag in die pom.xml zu Projekt hinzugefügt wird:

```

28
29 <plugin>
30   <groupId>org.codehaus.mojo</groupId>
31   <artifactId>cobertura-maven-plugin</artifactId>
32   <version>2.7</version>
33   <configuration>
34     <format>xml</format>
35     <maxmem>256m</maxmem>
36     <!-- aggregated reports for multi-module projects -->
37     <aggregate>true</aggregate>
38   </configuration>
39 </plugin>
40

```

Anpassen der Travis-Konfiguration

Damit coveralls.io die von Cobertura generierten Code-Coverage-Reports im Zuge des Deployment-Prozess von Travis auch übermittelt werden, muss die Travis Konfiguration .travis.yml durch folgenden Eintrag angepasst werden:

```

13
14 after_success:
15   - mvn clean cobertura:cobertura coveralls:report

```

Durch diesen Eintrag werden nach dem erfolgreichen Deployment die notwendigen Code-Coverage-Reports durch Cobertura generiert und mittels dem Coveralls-Maven-Plugin an coveralls.io gesendet.

Exkludieren von Code

Cobertura erlaubt das Ausschließen von Code auf drei unterschiedliche Arten. Zum einen kann ein Flag gesetzt werden welche dafür sorgt das „trivialer“ Code nicht in der Testabdeckung erfasst wird. Dadurch werden einfache Getter & Setter Methoden sowie Konstruktoren nicht berücksichtigt. Eine weitere Möglichkeit besteht in der Verwendung einer Ignore-Method-Annotation. Zuletzt ist es noch möglich komplette Dateien oder Packages zu exkludieren. Folgender Ausschnitt aus der pom.xml zeigt die Verwendung aller drei Möglichkeiten.

```
37      <!-- With the following lines you can exclude files from code coverage -->
38      <instrumentation>
39          <ignoreTrivial>true</ignoreTrivial>
40          <ignoreMethodAnnotation>Main.main.CoberturaIgnore</ignoreMethodAnnotation>
41          <ignores>
42              <ignore>com.example.boringcode.*</ignore>
43          </ignores>
44          <excludes>
45              <exclude>com/example/**/*Test.class</exclude>
46          </excludes>
47      </instrumentation>
48  </configuration>
```

Zur Verwendung der Ignore-Method-Annotation muss innerhalb des Quellcodes die Annotation via:

```
public @interface CoverageIgnore{}
```

Definiert werden. Um Code mittels dieser Annotation von der Erfassung der Testabdeckung auszuschließen muss der entsprechende Block (meist Methoden) durch die Annotation gekennzeichnet werden z.B.:

```
public class Foo {

    @CoverageIgnore
    public void start() {
        ...
    }

}
```

Verwenden von coveralls.io mittels Sage

Durch die in diesem Abschnitt beschriebenen Modifikationen wird gezeigt wie coveralls.io auf Basis des Code-Coverage-Tools Sage verwendet wird.

Saga

Bei Sage handelt es sich anders wie bei den Tools Jacoco und Cobertura nicht um ein Code-Coverage-Tool zur Ermittlung der Testabdeckung von Java Dateien, sondern es erfasst die Testabdeckung von JavaScript und HTML Dateien.

Inkludieren des Sage-Plugins

Wie auch bei den beiden vorherigen Code-Coverage-Tools existiert auch für Sage ein entsprechendes Maven-Plugin welches durch folgenden Eintrag in die pom.xml zum Projekt hinzugefügt wird:

```

28
29     <plugin>
30       <groupId>com.github.timurstrekalov</groupId>
31       <artifactId>saga-maven-plugin</artifactId>
32       <version>1.5.5</version>
33       <executions>
34         <execution>
35           <goals>
36             <goal>coverage</goal>
37           </goals>
38         </execution>
39       </executions>
40       <configuration>
41         <baseDir>http://localhost:${jasmine.serverPort}</baseDir>
42         <outputDir>${project.build.directory}/saga-coverage</outputDir>
43         <noInstrumentPatterns>
44           <pattern>./spec/*.*/</pattern>
45           <pattern>./classpath/*.*/</pattern>
46           <pattern>./webjars/*.*/</pattern>
47         </noInstrumentPatterns>
48       </configuration>
49     </plugin>
50

```

Anpassen der Travis-Konfiguration

Damit die von Saga erzeugten Code-Coverage-Reports beim Deployment-Prozess von Travis auch an coveralls.io gesendet werden muss die Travis Konfiguration `.travis.yml` durch folgenden Eintrag angepasst werden.

```

13
14   after_success:
15     - mvn clean test saga:coverage coveralls:report

```

Durch diesen Eintrag werden nach dem erfolgreichen Deployment die notwendigen Code-Coverage-Reports durch Saga generiert und mittels dem Coveralls-Maven-Plugin an coveralls.io gesendet.

Exkludieren von Code

Auch das Tool Saga ermöglicht es gewisse Teile des Codes von der Überprüfung auszuschließen. Dazu müssen folgende Zeile in die `pom.xml` eingetragen werden.

```

47     </noInstrumentPatterns>
48     <excludes>
49       **/set1/*-TestRunner.html,
50       **/set2/*-TestRunner.html
51     </excludes>
52   </configuration>

```

Ähnlich wie bei Jacoco können bei Saga einzelne Dateien von der Überprüfung ausgeschlossen werden, dazu müssen die einzelnen Dateien innerhalb des `<excludes> ... </excludes>` Elements welches innerhalb des `<configuration> ... </configuration>` Elements notiert werden muss, durch Komma getrennt angegeben werden.

Kombination mit anderen Code-Coverage-Tools

Es ist möglich Saga mit einem der anderen Code-Coverage-Tools zu kombinieren dazu müssen lediglich beide Maven-Plugins in die `pom.xml` eingetragen werden. So ist es möglich die Testabdeckung sowohl

für Java als auch JavaScript und HTML Dateien zu erfassen. Dazu muss eine der folgenden Travis Konfigurationen verwendet werden:

```
14  after_success:
15    - mvn clean test saga:coverage cobertura:cobertura coveralls:report
16
17  after_success:
18    - mvn clean test saga:coverage jacoco:report coveralls:report
```

Weiterführende Informationen

Weiterführenden Informationen findet ihr zum eine auf der github-Seite des Coveralls-Maven-Plugins:

<https://github.com/trautonen/coveralls-maven-plugin>

Informationen zu den einzelnen Code-Coverage-Tools findet ihr unter:

<http://www.eclemma.org/jacoco/>

<http://cobertura.github.io/cobertura/>

<http://timurstrekalov.github.io/saga/>

Zum anderen findet ihr auf meinem github-Account das Projekt coveralls-ready welches sämtliche Konfigurationen für alle Code-Coverage-Tools (pom.xml) enthält:

<https://github.com/ManuelBothner/se-coveralls-ready>