ToiletPaper #122

Remote Debugging von Maven (Surefire) Tests

Autor: Jan Machnik / Software Engineer/ Standort Stuttgart

✗ Problem

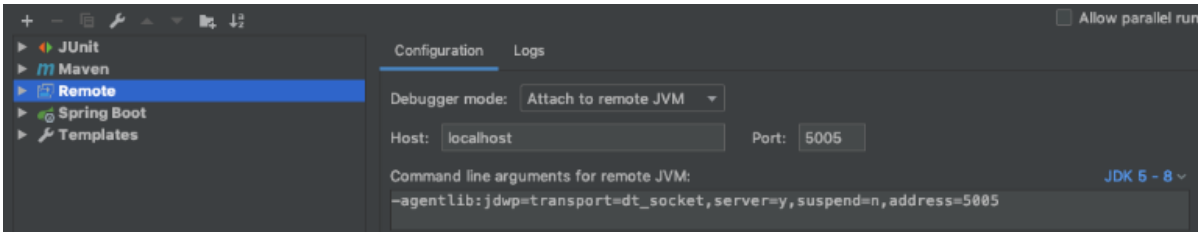
Vielleicht kennen auch andere das Phänomen, dass alle Unit-Tests in IntelliJ IDEA brav durchlaufen, aber Maven da ganz anderer Meinung ist. Wegen fehlgeschlagener Tests in eurer Buildpipeline werdet ihr mit einem „BUILD FAILURE, viel Spaß beim Suchen!“ belohnt. Aber wie bekommt man den Code eigentlich zur Buildzeit debugged? Einfach einen Breakpoint setzen und den Test in IntelliJ IDEAs Debug Modus nochmal ausführen führt hier nicht zum Ziel. Wir müssen den Test unter den gleichen Voraussetzungen starten, wie es Maven auch tut.

✓ Lösung

Mit Hilfe des Maven Surefire Plugin, das Maven während der Test-Phase des Build Lifecycle nutzt, können wir genau das machen. Mit dem Plugin ist es möglich Maven auf einen Port horchen zu lassen und sich auf diesen mit eurer IDE zu verbinden. Der Remote Debugger ist aber nicht nur auf lokale Maven-Projekte beschränkt. Auch Gradle-Projekte, JARs, Anwendungen auf anderen Servern und sogar Docker-Container lassen sich auf diese Weise debuggen.

➔ Beispiel

Zunächst müssen wir eine remote Run Configuration in IntelliJ IDEA erstellen (funktioniert ähnlich natürlich auch in anderen IDEs wie Eclipse, …), darin den Port festlegen (Maven nutzt standardmäßig Port 5005) und ein paar weitere Kommandozeilenparameter setzen. Eine Beispielkonfiguration kann dem Bild entnommen werden.



Anschließend können Breakpoints gesetzt und der Build-Prozess mit dem Kommandozeilenparameter -**Dmaven.surefire.debug** nochmal angestoßen werden.
Für einen einzelnen Test könnte der Befehl wie folgt aussehen: **mvn package -Dtest=TestKlasse.java#testMethode -Dmaven.surefire.debug**
Für alle Tests: **mvn package -Dmaven.surefire.debug**
Maven meldet sich, sobald es bei den Tests angekommen ist:

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Listening for transport dt_socket at address: 5005
```

Nun wird es Zeit, die vorhin erstellte Run Configuration in IntelliJ IDEA zu starten und zu hoffen, dass die Breakpoints richtig gesetzt sind ;). Ab diesem Zeitpunkt können wir wie gewohnt durch den Code steppen und die Ursache für unseren Fehler suchen. In dem Sinne, fröhliches Debuggen!

+ Weiterführende Aspekte

- Zur Dokumentation des Maven Plugins
- Allgemeinere Informationen zum Remote Debugging