01

Um teste falhando

Transcrição

Chegamos em um momento muito importante do treinamento, temos o nosso *job* configurado e executando o processo de *deploy* de forma automatizada, e mais do que isso, ele promove a construção continuamente, e isso é muito importante para nós, porque isso vai fazer com o que o Jenkins, o ambiente de integração contínua antecipe problemas, antes de termos dor de cabeça com eles.

Podemos lembrar do exemplo de configuração das notificações de e-mail, durante aquele exemplo, foi inserido e commitado no Github um erro de compilação no código fonte, e na hora em que o Jenkins executou o *job*, ele teve uma falha na construção. Obviamente, ele fez a notificação do problema por e-mail e nós ficamos sabendo desse erro antes que os desenvolvedores começassem a baixar esse código problemático, evitando que eles perdessem tempo tentando identificar o problema, quem e o quê o causou, para poder corrigi-lo. Logo, o problema foi antecipado.

Então, agora estamos focados no nosso código, na sua qualidade. Checar se o código compila ou não seria o nível mais básico de verificação de qualidade que o Jenkins pode fazer para nós. Uma outra coisa que o Jenkins pode analisar, é se os testes unitários estão funcionando corretamente, se não há nenhuma falha. Se alguém altera o código, e essa alteração faz algum teste falhar, o time de desenvolvedores gostaria de ser avisado dessa falha de forma proativa, e não esperar que essa funcionalidade lá na frente dê um erro em produção, ou que os próprios desenvolvedores tenham que executar os testes eles mesmos para verem que algum não passa e a nova funcionalidade não está funcionando como o esperado. Logo, o objetivo é que o Jenkins nos notifique disso também.

Sabemos que o Jenkins não executa os testes unitários, isso é responsabilidade do Maven. Durante a execução da *meta* `package`, ele irá passar por várias fases do ciclo de vida, e nós já vimos que ele passa pela fase de compilação, porque nós queremos compilar o código fonte que queremos empacotar. Mas ele também passa pela fase de testes, porque queremos garantir que os testes que temos no projeto estão passando, antes de gerarmos o pacote, nada mais justo, correto?

Podemos ver a saída de console do *job* `argentum-web` e verificar que durante a fase de testes unitários, o Maven chama o plugin `surefire` e executa todos os testes unitários do projeto, que ao todo são 13.

```
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ argentum-web ---
[INFO] Surefire report directory: /Users/Caelum/.jenkins/workspace/argentum-web/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
objc[2945]: Class JavaLaunchHelper is implemented in both
/Library/Java/JavaVirtualMachines/jdk1.8.0_51.jdk/Contents/Home/jre/bin/java and
/Library/Java/JavaVirtualMachines/jdk1.8.0_51.jdk/Contents/Home/jre/lib/libinstrument.dylib. One of the two will
be used. Which one is undefined.
Running br.com.caelum.argentum.indicadores.MediaMovelPonderadaTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.152 sec
Running br.com.caelum.argentum.indicadores.MediaMovelSimplesTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running br.com.caelum.argentum.modelo.CandlestickFactoryTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.017 sec
Running br.com.caelum.argentum.modelo.NegociacaoTest
Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.006 sec
Running br.com.caelum.argentum.reader.LeitorXMLTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.651 sec

Results :

Tests run: 13, Failures: 0, Errors: 0, Skipped: 0
```

Nós sabemos que se algum desses testes falha, o *build* do Maven irá falhar também, e consequentemente a construção global do Jenkins também irá falhar. Podemos testar isso modificando algum teste para ele falhar, commitar essa alteração e executar a construção no Jenkins.

```
Failed tests:   sequenciaSimplesDeCandles(br.com.caelum.argentum.indicadores.MediaMovelSimplesTest): expected:
<1.0> but was:<2.0>

Tests run: 13, Failures: 1, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 8.501 s
[INFO] Finished at: 2016-06-22T16:09:40-03:00
[INFO] Final Memory: 24M/205M
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.12.4:test (default-test) on
project argentum-web: There are test failures.
[ERROR]
[ERROR] Please refer to /Users/Caelum/.jenkins/workspace/argentum-web/target/surefire-reports for the individual
test results.
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
Build step 'Chamar alvos Maven de alto nível' marked build as failure
Sending e-mails to: soeirosantos@gmail.com
Finished: FAILURE
```

Vemos que o *build* é interrompido, encerrando a construção com erro, e um e-mail é enviado notificando essa falha. **Mas não podemos nos esquecer de modificar novamente o teste para que ele volte a funcionar.**

Executando novamente a construção no Jenkins, vemos que o *build* volta a funcionar e novamente somos notificados via e-mail, só que dessa vez avisando que o *build* voltou ao normal. Até aqui, não fizemos nenhuma alteração no Jenkins, porque como ele utiliza o Maven, que por sua vez, para empacotar o projeto, passa pela fase de testes, acabamos ganhando "de graça" esse recurso na verificação de qualidade do nosso código. Então, ganhamos a verificação de se o código compila ou não, de se os testes unitários estão falhando ou não, só que queremos um pouco mais do Jenkins, que ele execute um tipo de teste que não é executado de forma padrão pelo Mavem, os **testes funcionais**.