



Jenkins



Deze cursus is eigendom van de VDAB©

Inhoudsopgave

1	INLEIDING.....	3
1.1	Doelstelling	3
1.2	Vereiste voorkennis.....	3
1.3	Nodige software	3
1.4	Continuous Integration (CI).....	3
2	DOWNLOAD EN INSTALLATIE	4
2.1	Download	4
2.2	Installatie.....	4
2.3	Plugins.....	4
2.3.1	Git	4
2.3.2	Mail.....	5
2.4	Configuratie.....	5
2.4.1	JDK	5
2.4.2	Maven.....	5
2.4.3	E-mail	5
2.5	Extra plugins.....	6
2.5.1	Maven Integration	6
2.5.2	Warnings Next Generation.....	6
2.5.3	JaCoCo.....	6
2.5.4	Deploy to container	6
3	PROJECT	7
4	JOB	8
4.1	Maken	8
4.2	Uitvoeren	8
4.3	Resultaat	9
5	TEST.....	10
5.1	Test toevoegen.....	10
5.2	Job uitvoeren.....	10
5.3	Correctie	11

5.4	Job terug uitvoeren	11
6	E-MAIL.....	12
6.1	Job uitbreiden	12
6.2	Job uitvoeren.....	12
7	JACOCO	13
7.1	Extra code	13
7.2	pom.xml	13
7.3	Job uitbreiden	13
7.4	Job uitvoeren.....	14
7.5	Extra tests	14
8	SPOTBUGS.....	16
8.1	pom.xml	16
8.2	Job uitbreiden	16
8.3	Job uitvoeren.....	16
9	JAVADOC.....	18
9.1	pom.xml	18
9.2	Job uitbreiden	18
9.3	Job uitvoeren.....	18
10	CONTINUOUS DELIVERY	19
10.1	War ter beschikking stellen.....	19
10.2	War deployen op webserver.....	19
11	COLOFON	20

1 INLEIDING

1.1 Doelstelling

Je leert werken met Jenkins.

Dit is een open source CI (continuous integration) tool.

Je leert straks wat continuous integration betekent.

1.2 Vereiste voorkennis

- Java
- Spring
- Maven
- Git

1.3 Nodige software

- IntelliJ

1.4 Continuous Integration (CI)

Je ontwikkelt een enterprise applicatie zelden alleen, maar meestal met een team.

Een versiebeheersysteem (Git, ...) op een remote repository bevat de sources van de applicatie.

Je doet volgende stappen om een wijziging aan te brengen aan de applicatie:

1. Je pullt de sources van de remote repository naar je private repository.
2. Je doet de wijzigingen.
3. Je voert de unit tests uit die bij die wijzigingen horen.
4. Je voert de integration test uit die bij die wijzigingen horen.
5. Je commit de wijzigingen naar je private repository.
6. Je pusht je wijzigingen van je private repository naar de remote repository.

Na deze push moeten alle applicatie onderdelen (ook diegene die je niet wijzigde) correct werken.

Je controleert dit door alle unit tests en alle integration tests van de applicatie uit te voeren.

Dit vraagt veel tijd. Het is onhandig dit uit te voeren op de computer van een ontwikkelaar.

Een Continuous integration (CI) tool lost dit probleem op.

Je voert een CI tool uit op een test server.

Dit is een aparte computer, niet de computers van een ontwikkelaars.

De CI tool controleert regelmatig (bvb. om de minuut) of een ontwikkelaar een push deed naar de remote repository. Als dit gebeurt, compileert de tool de volledige applicatie en voert alle tests uit. De tool publiceert daarna een rapport van deze taken.

Wanneer het past voor de ontwikkelaars, lezen ze dit rapport.

Het bevat informatie over mislukte tests.

De ontwikkelaars kunnen daarmee correcties aanbrengen.

Handig is dus:

- De CI tool compileert en test *automatisch* nadat één van de ontwikkelaars een push doet. Je moet dit werk dus niet handmatig in de CI tool starten.
- De CI tool werkt op een aparte test server. De CI tool verbruikt geen processor of RAM van een computer van een ontwikkelaar.
- Terwijl de CI tool zijn werk doet, doen de ontwikkelaars ander werk. Ze wachten niet op de CI tool. Wanneer het hen past lezen ze het rapport van de CI tool. De CI tool kan dit rapport zelfs met een mail naar de ontwikkelaars sturen.

Een CI tool kan naast het compileren en testen van de applicatie extra taken uitvoeren, zoals:

- De Javadoc documentatie van de applicatie maken.
- De kwaliteit van de code van de applicatie controleren.

Je ziet dit verder in de cursus.

2 DOWNLOAD EN INSTALLATIE

Jenkins draait normaal op een aparte test server. Je voert in deze cursus Jenkins uit op je computer. Dit is waarschijnlijk ook de computer waarop je ontwikkelt.

2.1 Download

Je kan Jenkins downloaden op <https://jenkins.io/>, met de knop Download.

Je kiest daarna onder Long-Term Support (LTS) voor Generic Java package (.war).

Je krijgt dan een bestand jenkins.war

2.2 Installatie

De extensie van dit bestand is war. Jenkins is dus een web applicatie.

Je kan die op een webserver (bijvoorbeeld Tomcat) installeren.

Je kan daarna naar de Jenkins web applicatie surfen om taken op Jenkins uit te voeren.

Er bestaat ook een andere, eenvoudige manier om Jenkins uit te voeren.

jenkins.war bevat een embedded (ingebakken) webserver.

Als je jenkins.war start via een console venster, start deze embedded webserver.

Je kan daarna met een browser naar die webserver surfen om taken op Jenkins uit te voeren.

1. Maak in de root van de harde schijf een folder jenkins.
2. Kopieer jenkins.war in die folder.
3. Start een Command Line venster.
4. Typ `cd /jenkins`. Druk Enter. Je bevindt je in de folder jenkins.
5. Typ `java -jar jenkins.war -httpPort=80811`. Druk Enter.
Je start zo Jenkins. Die bevat een embedded webserver die op TCP/IP poort 8081.
Het starten duurt even. Op het einde zie je: `INFO: Jenkins is fully up and running`.
Laat het Command Line venster open, anders zou je Jenkins sluiten.

Jenkins toont teksten in de taal (Engels, Nederlands, ...) die je in de browser opties configureerde.

Deze cursus gebruikt Engels. We raden daarom aan in je browser ook Engels te kiezen.

Surf met een browser naar `localhost:8081`. Je ziet de startpagina van Jenkins.

Typ het paswoord van de Jenkins administrator.

Je krijgt een hint in welk bestand je dit paswoord vindt.

Kies daarna Continue.

2.3 Plugins

Jenkins bevat standaard een beperkte functionaliteit.

Je voegt functionaliteit toe door plugins toe te voegen aan Jenkins.

Kies `Select plugins to install`.

Jenkins stelt in de volgende pagina voor enkele plugins te installeren. Die zijn niet allemaal nuttig.

Kies daarom op None.

Je voegt zelf de plugins toe die je nodig hebt in deze cursus.

2.3.1 Git

Je werkt met de Git plugin samen met een project op een Git repository.

- Typ in het tekstvak boven in de pagina `git`.
Je ziet een lijst met de plugins waarvan in de naam het woord `git` voorkomt.
- Plaats een vinkje bij Git.

2.3.2 Mail

Je stuurt met de Mailer plugin mails naar je medewerkers als het compileren of testen van jullie project mislukt.

- Typ in het tekstvak boven in de pagina mailer.
Je ziet een lijst met de plugins waarvan in de naam het woord mailer voorkomt.
- Plaats een vinkje bij Mailer.

Kies onder in de pagina Install. Jenkins installeert de plugins. Dit duurt een tijdje.

Maak in de volgende pagina een gebruiker. Kies daarna Save and Continue.

Kies in de volgende pagina Save and Finish.

Kies in de volgende pagina Start using Jenkins.

Je komt daarna op de hoofdpagina van Jenkins.

2.4 Configuratie

2.4.1 JDK

Je geeft aan in welke directory de JDK (Java Development Kit) geïnstalleerd is:

1. Kies Manage Jenkins (links in de pagina).
2. Kies Global Tool Configuration.
3. Kies Add JDK.
4. Typ een vrij te kiezen naam bij Name.
5. Verwijder het vinkje bij Install automatically.
6. Typ bij JAVA_HOME het pad naar directory waar je JDK geïnstalleerd is.
Verlaat het veld. Wacht enkele seconden. Als het pad verkeerd is, zie je een fout.

2.4.2 Maven

Je installeert Maven:

1. Kies Add Maven.
2. Typ een vrij te kiezen naam bij Name.
3. Kies Save.

2.4.3 E-mail

Je configureert de mailserver en de account waarmee Jenkins mails verstuurt.

Je gebruikt normaal de mailserver van je firma. Je zal hier de mailserver van GMail gebruiken:

1. Kies Manage Jenkins (links in de pagina).
2. Kies Configure System.
3. Typ de gebruikersnaam van je GMail account bij System Admin e-mail address (onder Jenkins Location).
4. Typ smtp.gmail.com bij SMTP server (onder E-mail notification).
5. Kies Advanced.
6. Plaats een vinkje bij Use SMTP Authentication.
7. Typ de gebruikersnaam van je GMail account bij punt 3 bij User Name.
8. Typ het bijbehorende paswoord bij Password.
9. Plaats een vinkje bij Use SSL.
10. Typ 465 bij SMTP Port.
11. Plaats een vinkje bij Test configuration by sending test e-mail.
12. Typ een e-mail adres bij Test e-mail recipient.
13. Kies Test configuration.
De boodschap Email was succesfully sent verschijnt.
14. Kies Save.

De in-box van de gebruiker, vermeld bij puntje 9, bevat een mail met de titel Test email #1.

2.5 Extra plugins

Je installeert als volgende enkele extra Jenkins plugins die je zal nodig hebben in deze cursus.

2.5.1 Maven Integration

Deze plugin zorgt voor een optimale integratie tussen Jenkins en Maven.

1. Kies Manage Jenkins (links in de hoofdpagina).
2. Kies Manage Plugins.
3. Kies Available.
4. Typ maven naast Filter (rechts boven).
5. Plaats een vinkje bij Maven Integration.

2.5.2 Warnings Next Generation

Deze plugin controleert de kwaliteit van je code (verder in de cursus).

1. Typ warnings next generation naast Filter (rechts boven).
2. Plaats een vinkje bij Warnings Next Generation.

2.5.3 JaCoCo

Deze plugin controleert de kwaliteit van je tests (verder in de cursus).

1. Typ jacoco naast Filter (rechts boven).
2. Plaats een vinkje bij JaCoCo.

2.5.4 Deploy to container

Deze plugin depolyt een war bestand op een webserver (verder in de cursus).

1. Typ deploy to container naast Filter (rechts boven).
2. Plaats een vinkje bij Deploy to container.
3. Kies Install without restart.
4. Wacht tot alles geïnstalleerd is.

3 PROJECT

Het project waarmee je Jenkins leert kennen is het Maven project GoedeDoe1.

Je vindt dit in het materiaal bij deze cursus.

Je importeert het project in IntelliJ:

1. Kies in het menu `File` de opdracht `Open`.
2. Kies `pom.xml` in de map `GoedeDoe1`. Kies `OK`.
3. Kies `Open as Project`.

Maak van dit project een Git project (zie Git cursus).

Doe een eerste commit.

Push naar een remote public repository op GitHub.

4 JOB

Je maakt in Jenkins één job per project dat je met Jenkins wil verwerken (compileren, testen, ...)

4.1 Maken

Je maakt een job voor het project GoedeDoel:

1. Kies links in de Jenkins hoofdpagina New Item (of je kiest create new jobs).
2. Typ bij Enter an item name een naam voor de job: Goede doel.
Deze naam mag verschillen van de naam van de Maven applicatie.
3. Kies Maven project.
4. Kies OK.

Je komt op een pagina waarin je de detail van de job definieert.

1. Kies bij Source Code Management voor Git.
2. Typ bij Repository URL de URL van je remote repository. Verlaat het veld.
Wacht enkele seconden. Als de URL verkeerd is, zie je een foutmelding.
3. Plaats bij Build Triggers een vinkje bij Poll SCM. Jenkins zal dan op regelmatige basis controleren of je een nieuwe versie van het project pushte naar je remote repository.
Typ daarna H/2 * * * * * bij Schedule.

Je geeft daarmee aan hoe regelmatig Jenkins dit controleert.

- H/2 iedere 2 minuten
- * van ieder uur
- * van iedere dag
- * van iedere maand
- * van ieder jaar


4. Kies Save.

4.2 Uitvoeren

Als je 2 minuten wacht, voert Jenkins de job uit:

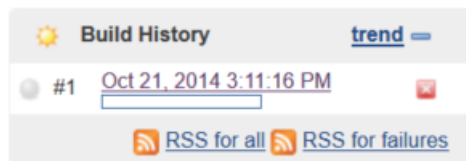
je definieerde in de job dat Jenkins om de 2 minuten de remote repository naziet op wijzigingen.

Je wacht niet zolang: je start de job manueel.

Kies daartoe links Build Now of het icoon 

Je zal dit icoon op veel pagina's zien. Je kan het altijd gebruiken om de job manueel te starten.

Je ziet na enkele seconden links een balk. Deze geeft de de vooruitgang van de job aan:









Wanneer de balk verandert in een hyperlink, is de job uitgevoerd.

4.3 Resultaat

Kies links boven Jenkins om de beginpagina te zien.

Je ziet in het midden van de pagina een samenvatting van de uitgevoerde job:

S	W	Name ↓	Last Success	Last Failure	Last Duration	
		Goede doel	4 min 32 sec - #1	N/A	14 sec	

- De 1° kolom geeft informatie over de *meest recente* uitvoering van de job.
 -  uitgevoerd zonder fouten
 -  slecht uitgevoerd (slecht geconfigureerde job of compiler fouten)
 -  foutieve testen of (later in de cursus) code van mindere kwaliteit
- De 2° kolom geeft informatie over *alle* uitvoeringen van de job (niet enkel de meest recente uitvoering).
 - Naarmate je een job meer lukt, zie je een mooier weer icoon ()
 - Naarmate je een job meer mislukt, zie je een slechter weer icoon (, )

Kies in de 3° kolom [Goede doel](#). Je ziet een pagina met details over de uitvoeringen van de job.

Kies [Latest Test Result](#). Je ziet een rapport met de resultaten van de JUnit tests van het project:

Test Result

0 failures

				1 tests
Module	Fail	(diff)	Total	(diff)
be.vdab:GoedeDoel	0		1	+1

De tekst 0 failures met daaronder een blauwe balk geeft aan dat alle tests slaagden.

Kies links Console Output. Je ziet de output van het uitvoeren van de job.

Dit is interessant als die uitvoering mislukt. Je ziet dan foutmeldingen in deze pagina.

Deze foutmeldingen helpen je de fout op te sporen te corrigeren.

5 TEST

Je zal aan het project een falende test toevoegen. Je leert hoe Jenkins hiermee omgaat.

5.1 Test toevoegen

Voeg een method toe aan de class GoedeDoelTest.

Die controleert of een nieuw doel € 0 heeft opgebracht.

```
@Test
void eenNieuwDoelHeeftNogGeenOpbrengst() {
    assertThat(doe1.getOpbrengst()).isZero();
}
```

Deze test zal falen met een NullPointerException:

in de class GoedeDoel wordt de variabele opbrengst niet geïnitieerd.

Commit de uitbreiding. Push naar je remote repository.

5.2 Job uitvoeren

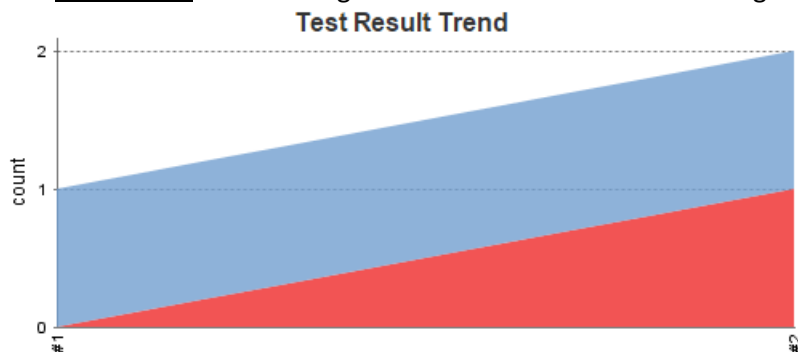
Voer in Jenkins de job Goede doel opnieuw uit.

Je ziet daarna in de welkompagina volgend resultaat:

S	W	Name ↓	Last Success	Last Failure	Last Duration	
		Goede doel	21 sec - #2	N/A	8,4 sec	

 wijst op één of meerdere falende tests.

Kies [Goede doel](#). Je ziet een grafiek met statistieken over de uitgevoerde JUnit tests:



#1 (onder de grafiek) staat voor de 1° uitvoering van de job.

Hierbij werd één test uitgevoerd (count 1) en die had succes (blauwe kleur).

#2 (onder de grafiek) staat voor de 2° uitvoering van de job.

Hierbij werden twee tests uitgevoerd (count 2). Één had succes (blauwe kleur) en één mislukte.

Kiest Latest Test Results. Je ziet volgende pagina:

Test Result

1 failures (+1)

2 tests (+1)			
Module	Fail	(diff) Total	(diff)
be.vdab.goededoel	1	+1	2 +1

Failed Tests

[be.vdab.goededoel](#)

Test Name	Duration	Age
+ be.vdab.entities.GoedeDoelTest.eenNieuwDoelHeeftNogNietsOpgebracht	0.003	1

Je ziet met [+](#) detail informatie over de mislukte test:

Test Name	Duration	Age
be.vdab.goededoel.domain.GoedeDoelTest.eenNieuwDoelHeeftNogGeenOpbrengst		
Error Details Expecting actual not to be null + Stack Trace	84 ms	1

5.3 Correctie

Corrigeer de fout met een aanpassing in GoedeDoel:

```
private BigDecimal opbrengst = BigDecimal.ZERO;
```

Commit de uitbreiding. Push naar je remote repository.

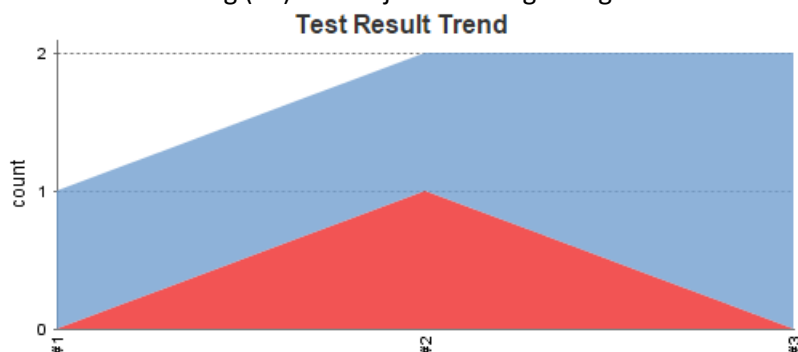
5.4 Job terug uitvoeren

Voer in Jenkins de job opnieuw uit. Je ziet daarna in de welkompagina volgend resultaat:

S	W	Name ↓	Last Success	Last Failure	Last Duration
		Goede doel	16 sec - #3	N/A	12 sec 

Kies Goede doel. Je ziet terug de grafiek met statistieken over de uitgevoerde JUnit tests.

De derde uitvoering (#3) van de job bevat 2 geslaagde tests:



6 E-MAIL

Nadat Jenkins een job uitvoerde, kan Jenkins een e-mail sturen naar één of meerdere personen. Dit zijn normaal personen die meewerken aan het project.

Jenkins stuurt een e-mail

- als de uitvoering van een job mislukt
- als de uitvoering van een job lukt nadat die bij de vorige uitvoering mislukte

6.1 Job uitbreiden

Je breidt in Jenkins de job Goede doel uit, zodat hij een e-mail verstuurt:

1. Kies in de beginpagina in het midden Goede doel.
2. Kies links Configure.
3. Plaats een vinkje bij E-mail notification.
4. Typ één of meerdere e-mail adressen bij Recipients, gescheiden door een spatie.
5. Kies Save.

6.2 Job uitvoeren

Wijzig een unit test zodat die mislukt.

Voer in Jenkins de job Goede doel opnieuw uit.

Je ziet daarna in de in-box van een recipient een mail met het rapport van de uitgevoerde job.

Corrigeer de unit test. Voer de job opnieuw uit.



Opmerking: Andere plugins kunnen op andere manieren personen waarschuwen dat een job is uitgevoerd. Er bestaat een plugin om een SMS te versturen, een plugin om een boodschap te tonen rechts in de taakbalk van de computer van de persoon, ...

7 JACOCO

Een code coverage tool controleert welke code (bvb. uit de class GoedeDoel) doorlopen wordt bij het uitvoeren van de bijbehorende test (bvb. GoedeDoelTest) en maakt hiervan een rapport.

Als de tests bepaalde code helemaal niet, of slechts gedeeltelijk doorlopen, is dit een indicator dat je voor die code extra tests moet schrijven.

Er bestaan meerdere Java code coverage tools. JaCoCo is er één van.

7.1 Extra code

Voeg code toe aan GoedeDoel. Je schrijft geen unit test voor deze code.

```
@Override
public boolean equals(Object object) {
    if (object instanceof GoedeDoel) {
        var ander = (GoedeDoel) object;
        return naam.equalsIgnoreCase(ander.naam);
    }
    return false;
}
```

7.2 pom.xml

Jenkins voert JaCoCo uit als een stap van het Maven build process van je project.

Voeg de Maven plugin voor JaCoCo toe aan pom.xml, voor de regel </plugins>:

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.5</version>
  <executions>
    <execution>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
    </execution>
    <execution>
      <id>report</id>
      <phase>prepare-package</phase>
      <goals>
        <goal>report</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Commit de wijziging. Push naar je remote repository.

7.3 Job uitbreiden

Het is niet omdat pom.xml de JaCoCo plugin bevat, dat Jenkins hem default uitvoert.

Je breidt in Jenkins de job Goede doel uit, zodat hij na de build het JaCoCo rapport toont:

1. Kies in de beginpagina in het midden Goede doel.
2. Kies links Configure.
3. Kies Add post-build action bij Post-Build Actions en kies Record JaCoCo coverage report.
4. Kies Save.

7.4 Job uitvoeren

Voer de job opnieuw uit.

Je ziet in het resultaat van de build een statistiek over het uitvoeren van Jacoco:



Jacoco - Overall Coverage Summary

INSTRUCTION	21%	<div><div></div></div>
BRANCH	0%	<div><div></div></div>
COMPLEXITY	25%	<div><div></div></div>
LINE	33%	<div><div></div></div>
METHOD	27%	<div><div></div></div>
CLASS	25%	<div><div></div></div>

- 25% van de Java byte code instructies werd getest.
- Geen enkele if of switch werd getest.
- Complexity valt buiten het bereik van de cursus.
- 38% van de source lijnen werd getest.
- 30% van de methods werd getest.
- 25% van de classes werd getest.



Het is niet de bedoeling dat je op alle items 100% haalt.

Je source bevat eenvoudige code, zoals getters en setters aangemaakt door je IDE.

Je schrijft hiervoor geen unit tests. Zo zakken de percentages van JaCoCo.

Kiest link in de pagina Coverage Report. Je ziet details van het JaCoCo rapport.

Kies daarin `be.vdab.goededoel.domain` en daarin `GoedeDoel`

Je ziet roze blokken in de code van de method `equals`.

Dit betekent dat deze code niet uitgevoerd werd tijdens de unit test

```
if (object instanceof GoedeDoel) {
    var ander = (GoedeDoel) object;
    return naam.equalsIgnoreCase(ander.naam);
}
return false;
```

7.5 Extra tests

Voeg testen toe aan `GoedeDoelTest`:

```
@Test
void doelenMetDezelfdeNaamZijnGelijk() {
    assertThat(doe1).isEqualTo(new GoedeDoel(NAAM));
}
@Test
void doelenMetVerschillendeNaamZijnVerschillend() {
    assertThat(doe1).isNotEqualTo(new GoedeDoel("WWF"));
}
```

Doe een commit. Push naar je remote repository.

Voer in Jenkins de job terug uit.

De statistiek over het uitvoeren van JaCoCo is verbeterd:



Jacoco - Overall Coverage Summary

INSTRUCTION	38%	<div><div></div></div>
BRANCH	50%	<div><div></div></div>
COMPLEXITY	33%	<div><div></div></div>
LINE	50%	<div><div></div></div>
METHOD	36%	<div><div></div></div>
CLASS	25%	<div><div></div></div>

Je hebt meer instructions, branches, complexity lines en methods getest.

Kies links in de pagina Coverage Report. Je ziet details van het JaCoCo rapport.

Kies daarin `be.vdab.goededoel.domain` en daarin `GoedeDoel`

Je ziet dat je een deel van de method `equals` niet test:

```
if (object instanceof GoedeDoel) {
    var ander = (GoedeDoel) object;
    return naam.equalsIgnoreCase(ander.naam);
}
return false;
```

Voeg code toe die dit verhelpt in `GoedeDoelTest`:

```
@Test
void doelVerschiltVanEenObjectMetEenAnderType() {
    assertThat(doel).isNotEqualTo(BigDecimal.ZERO);
}
```

Doet een commit. Push naar je remote repository.

Voer in Jenkins de job terug uit.

De class `GoedeDoel` is 100 % getest.

De pagina met het project bevat rechts ook een grafiek "Code Coverage Trend".

Deze toont de trend over de builds heen van aantal lijnen wel getest en aantal lijnen niet getest.

8 SPOTBUGS

Een static code analysis tool controleert of je project geen fouten bevat die veel voorkomen:

- strings vergelijken met == in plaats van met equals
- database connecties vergeten te sluiten
- ...

Zo'n tool bevat een database met gekende fouten. De tools controleert je code ten opzichte van deze fouten. Fouten die jij ook maakte komen in een rapport terecht.

SpotBugs is zo'n tool en is open source.

8.1 pom.xml

Voeg de Maven plugin voor SpotBugs toe aan pom.xml, voor de regel </plugins>:

```
<plugin>
  <groupId>com.github.spotbugs</groupId>
  <artifactId>spotbugs-maven-plugin</artifactId>
  <version>4.0.0</version>
</plugin>
```

Commit de wijziging. Push naar je remote repository.

8.2 Job uitbreiden

Het is niet omdat pom.xml de SpotBugs plugin bevat, dat Jenkins hem default uitvoert.

Je breidt in Jenkins de job Goede doel uit. Hij zal als een onderdeel van het Maven build process ook SpotBugs uitvoeren, via de plugin die je aan pom.xml toevoegde.

1. Kies in de beginpagina in het midden Goede doel.
2. Kies links Configure.
3. Typ clean install spotbugs:spotbugs bij Goals and options.
Het uitvoeren van SpotBugs wordt zo een extra stap van het Maven build proces.
4. Kies Add post-build action bij Post-Build Actions
en kies Record compiler warnings and static analysis result.
5. Kies SpotBugs bij Tool.
6. Kies Save.

8.3 Job uitvoeren

Voer in Jenkins de job Goede doel opnieuw uit.

De pagina met de job bevat links een onderdeel SpotBugs warnings. Kies dat.

Kies daarna in de pagina GoedeDoel.java

Je ziet een fout:

be.vdab.goededoel.domain.GoedeDoel defines equals and uses Object.hashCode()

Klik op de fout. Je ziet meer detail. Je hebt de method equals overridden, maar niet de method hashCode. Je verbreekt zo de regel dat als twee objecten volgens equals gelijk zijn, zij dezelfde returnwaarde moeten hebben van de method hashCode.

Voeg code toe die dit verhelpt in de class GoedeDoel:

```
@Override
public int hashCode() {
    return this.naam.toUpperCase().hashCode();
}
```

Voeg een bijbehorende test toe aan GoedeDoelTest:

```
@Test
void gelijkeDoelGevenHebbenDezelfdeHashCode() {
    assertThat(doe1).hasSameHashCodeAs(new GoedeDoel(NAAM));
}
```

Doet een commit. Push naar je remote repository.

Voert in Jenkins de job terug uit.

De pagina met de job bevat links een onderdeel SpotBugs warnings. Kies dat.

Je ziet dat er geen fouten meer zijn: No issues have been reported.

Je ziet ook een grafiek over het uitvoeren van de jobs heen.



Opmerking: naast SpotBugs bestaan nog tools die de kwaliteit van je code controleren en fouten zoeken: Checkstyle, Coverity, PMD, Sonargraph, ...

Je kan al deze tools integreren in Jenkins.

9 JAVADOC

Een onderdeel van een Jenkins job kan het genereren van project documentatie zijn.

Deze documentatie bestaat uit HTML pagina's met dezelfde lay-out als de Java API documentatie

Jenkins genereert deze documentatie op basis van commentaar die je aan de te documenteren classes toevoegt.

Schrijf de commentaar juist voor de class GoedeDoel:

```
/**
 * Een <strong>goed doel</strong> waarvoor men geld inzamelt
 * @author Joe Dalton
 */
```

Typ voor de constructor:

```
/**
 * Maakt een GoedeDoel object
 * @param naam De naam van het goede doel
 */
```

Typ voor de method getNaam:

```
/**
 * Geeft de naam terug
 * @return de naam
 */
```

Typ voor de method getOpbrengst:

```
/**
 * Geeft de opbrengst terug
 * @return de opbrengst
 */
```

9.1 pom.xml

Jenkins zal op basis van de commentaar de documentatie in HTML formaat maken als een stap van het Maven build process van je project.

Voeg aan pom.xml de Maven plugin voor javadoc toe, voor de regel </project>:

```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-javadoc-plugin</artifactId>
      <version>3.1.1</version>
    </plugin>
  </plugins>
</reporting>
```

Doe een commit. Push naar je remote repository.

9.2 Job uitbreiden

Je breidt in Jenkins de job Goede doel uit. Hij zal als een onderdeel van het Maven build process ook de documentatie maken, via de plugin die je aan pom.xml toevoegde:

1. Kies in de beginpagina in het midden Goede doel.
2. Kies links Configure.
3. Typ als extra javadoc:aggregate bij Goals and options.
Het aanmaken van documentatie wordt zo een stap van het Maven build proces.
4. Kies Save.

9.3 Job uitvoeren

Voer in Jenkins de job Goede doel opnieuw uit.

De pagina met het project bevat links een hyperlink Javadoc. Kies deze en daarna GoedeDoel.

10 CONTINUOUS DELIVERY

De stappen die je job tot nu toe uitvoert vallen onder de naam continuous integration.

Continuous delivery gaat nog een stap verder na continuous integration.

Als de job met succes is uitgevoerd, wordt het eindproduct geïnstalleerd

- op de test webserver waar testers de website vanuit hun browser testen
- of zelfs op de productie webserver
zodat de eindgebruikers de nieuwe versie van de applicatie gebruiken.

10.1 War ter beschikking stellen

Als eerste stap stel je het war bestand, dat het resultaat is van een geslaagde job uitvoering, ter beschikking van de beheerders van de test servers of productieservers.

Zij kunnen dit war bestand dan installeren op de test servers of productieservers.

Dit is nog niet de *automatische* continuous delivery, want zij installeren het bestand *handmatig*.

Je breidt in Jenkins de job Goede doel uit. Hij houdt het war bestand ter beschikking:

1. Kies in de beginpagina in het midden Goede doel.
2. Kies links Configure.
3. Voeg een spatie en package toe bij Goals and options.
4. Kies Add post-build action.
5. Kies Archive the artifacts.
6. Typ `**/*.war` bij Files to archive.
Dit betekent het bestand met de extensie war (`*.war`) in om het even welke directory die zelf een subdirectory kan zijn (`**`).
7. Kies Save.

Voer in Jenkins de job Goede doel opnieuw uit.

De pagina met de job bevat een hyperlink Goededoel-0.0.1-SNAPSHOT.war

De beheerders van de test servers of productieservers kunnen hiermee de gebouwde applicatie downloaden en installeren op hun servers.

10.2 War deployen op webserver

Start Tomcat standalone (via `startup.bat`) (niet via je IDE) op poort 8080.

Je breidt in Jenkins de job Goede doel uit. Hij installeert het war bestand op die Tomcat:

1. Kies in de beginpagina in het midden Goede doel.
2. Kies links Configure.
3. Kies Add post-build action.
4. Kies Deploy war/ear to a container.
5. Typ `**/*.war` bij WAR/EAR files.
6. Typ `goededoel` bij Context path.
Dit is het context path waarmee het war bestand op de Tomcat geïnstalleerd wordt.
7. Kies Add Container.
8. Kies Tomcat 9.x Remote.
9. Kies Add bij Credentials.
10. Kies Jenkins.
11. Typ `cursist` bij Username en bij Password. Kies Add.
12. Kies `cursist/*****` bij Credentials
13. Typ `http://localhost:8080` bij Tomcat URL.
14. Kies Save.

Voer de job Goede doel opnieuw uit.

Surf naar <http://localhost:8080/goededoel>. Je ziet de applicatie die op Tomcat draait.

11 COLOFON

Domeinexpertisemanager:	Jean Smits
Moduleverantwoordelijke:	Hans Desmet
Medewerkers:	Hans Desmet
Versie:	19/5/2020
Nummer dotatielijst:	