Nummer 3 - 2020

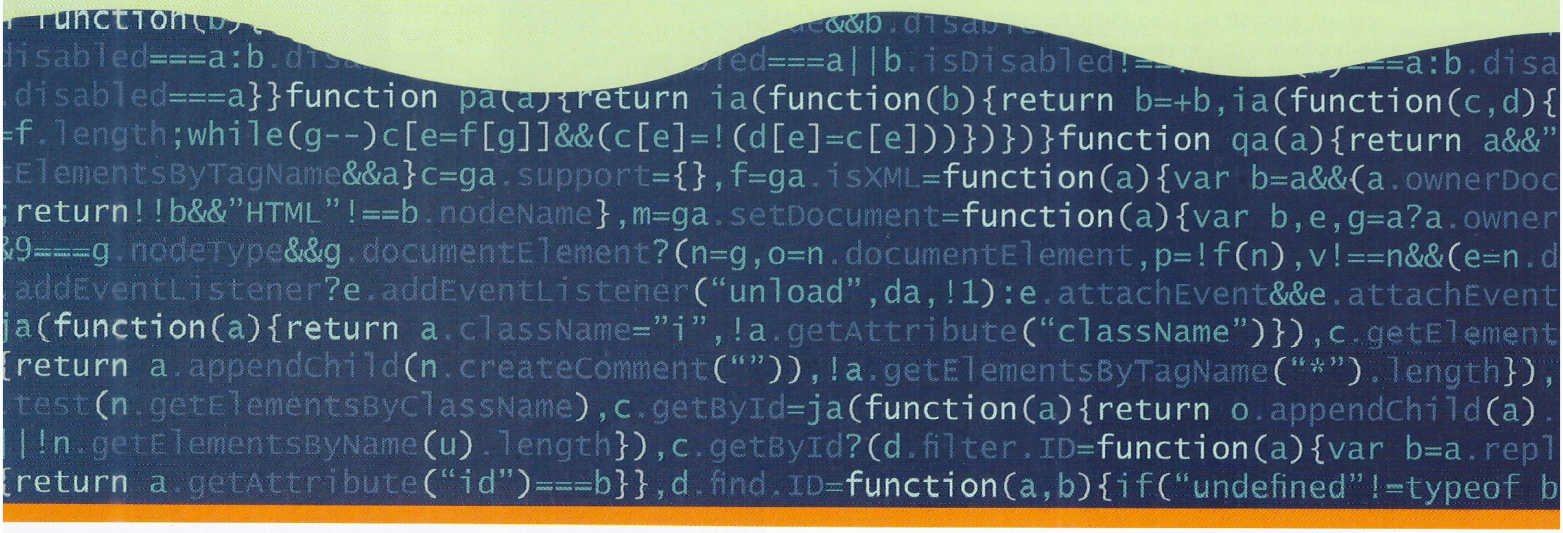
# JAVA MAGAZINE

Onafhankelijk tijdschrift voor de Java-profes

.nl. jug

# Take a deep dive!

## Into the code



## Features

**Actief worden** in open source

**JUnit5 migratie:** lessons learned

**Java 15** What's new?

# Actief worden in open source

Stel, je wil actief worden in open source. Wat komt daarbij kijken? Danny van Bruggen, oud-maintainer van JavaParser[1], en Jan Ouwens, maintainer van EqualsVerifier[2], geven een kijkje in de keuken.

In de eerste plaats gaat open source om code: iets waar alle lezers van dit blad van houden. Maar open source is meer dan dat.
Werken aan open source is vergelijkbaar met de gemiddelde IT-klus. Je werkt samen met je collega's en je bent min of meer op de hoogte van wat iedereen doet en waar het project heen gaat. Het grote verschil is dat alles op basis van vrijwilligheid gebeurt. Veel mensen willen gewoon een issue melden of oplossen zonder verder sociaal contact. Sommige mensen hebben wilde plannen en duiken diep in de materie, beginnen aan een groot project, maar geven dat halverwege op omdat de interesse weg is. Sommige mensen vinden juist de community erg interessant en blijven lang actief, maar leveren uiteindelijk niet zoveel op. Met deze groep mensen gaat het project zeker vooruit, maar krijg je datgene waar "Apache" zijn naam aan zou ontlenen: "a patchy server". Een project met goede bedoelingen in de kern, en vijf lagen pleisters eroverheen.
De jackpot is de persoon die diep de materie ingaat en ook daadwerkelijk structurele verbeteringen doorvoert en een deel van de pleisters verwijdert.

## Motivatie

Voordat je begint, is het goed om bij jezelf na te gaan waarom je actief wil worden in open source. Er zijn namelijk meerdere goede redenen die beïnvloeden hoe je dit het beste doet. Wil je meer leren over techniek? Wil je leren een project te managen? Wil je netwerken met gepassioneerde developers over de hele wereld? Of wil je gewoon je cv boosten?
Er zijn helaas weinig bedrijven die mensen de tijd geven om aan open source te werken, dus houd er rekening mee dat je dit in je vrije tijd moet doen. En ondanks initiatieven als Tidelift en GitHub Sponsors, waarmee mensen een soort betaald abonnement kunnen nemen op opensource projecten, zal je er waarschijnlijk geen geld mee verdienen. Maar het kan wel heel leuk zijn om eens je tanden te zetten in iets waar je op je werk misschien niet mee in aanraking komt, en het is bevredigend om te merken dat mensen aan de andere kant van de wereld gebruikmaken van iets wat jij gemaakt hebt.

## Hoe begin je?

Maar hoe begin je nu? Er zijn meerdere manieren om actief te worden, afhankelijk van je motivatie en van hoeveel tijd je wil besteden. Om te beginnen moet je een project kiezen. Kies iets wat je al kent omdat je het op je werk gebruikt, of zoek er een uit een lijst van projecten die hulp nodig hebben[3]. Je kunt ook zelf een project starten: daarover later meer. Als je weet aan welk project je wil werken, kun je de volgende stap zetten.

## Issues melden

De eenvoudigste manier om te werken aan open source, is het melden van een issue. Als je een probleem hebt gevonden in een library die je gebruikt, kun je dit aanmelden in de issue tracker van het project. Op die manier draag je bij aan het verbeteren van het project, omdat hieruit vaak een bugfix, een update van de documentatie, of zelfs een nieuwe feature volgt. Let er bij het aanmelden van een bug wel op dat je een gedegen reproductierecept aanbiedt. Als een project een invultemplate heeft bij het aanmelden van een issue, probeer dit dan ook zo goed mogelijk te volgen. Hiermee help je de main-

**Danny van Bruggen** is senior Java-ontwikkelaar bij Ordina. Hij heeft vijf jaar lang het JavaParser project onderhouden.

**Jan Ouwens** is senior Scala-ontwikkelaar bij Ordina en houdt zich bezig met Java en Scala. In 2009 is hij gestart met EqualsVerifier

tainer van het project enorm: hij of zij kan je op die manier sneller helpen. Bedenk dat maintainers ook mensen zijn zoals jij, die dit vaak in hun beperkte vrije tijd doen.

## Documentatie

Zoals we allemaal weten, hebben niet alle developers even veel zin in het schrijven en up-to-date houden van documentatie. Als jij hier wel goed in bent, is dit iets waar je een heel nuttige bijdrage in kan leveren. Ben je nog niet zo bekend met het project, dan kun je simpelweg de bestaande documentatie lezen en verbeteren. Je kunt bijvoorbeeld spel- en taalfouten corrigeren.

Als je het project beter kent, kun je ook inhoudelijke verbeteringen aanbrengen als je bijvoorbeeld een fout aantreft, of je kunt de tekst begrijpelijker maken waar nodig. Ken je het project op je duimpje? Dan zou je bijvoorbeeld een tutorial kunnen schrijven voor de website, of Javadoc toevoegen aan classes die dat niet hebben.

## Issues oplossen

Een andere manier om te beginnen, is door te kijken naar openstaande issues. Misschien zit er een vraag tussen waar je het antwoord op weet. In dat geval kun je dat natuurlijk geven: zo help je de vraagsteller, maar ook de maintainer die het issue zelf niet meer hoeft uit te zoeken.

Triage van open issues is ook een dankbare taak. Lees de open issues en vraag de aanmelders om meer informatie waar nodig, identificeer duplicates, koppel de juiste tags aan een issue: allemaal waardevolle bijdragen. Mocht je liever de code in willen duiken, dan kun je proberen om een simpele bug te fixen of een kleine feature toe te voegen. Veel projecten op GitHub gebruiken in hun issue trackers labels zoals "Help wanted" of "Easy". Issues met die labels zijn vaak goede plekken om hiermee te starten.

## Bouw een feature

Is er een feature die je mist, en jeuken je vingers? Bouw die dan gewoon zelf! Maar let op, hier zitten wat haken en ogen aan die je vooraf misschien niet verwacht. Het eerste dat je moet doen als je een grote feature wil gaan bouwen, is even contact opnemen met de maintainer van het project. Het meest demotiverende dat je kan overkomen is dat een pull request, waar je je hart en ziel in gestoken hebt, wordt geweigerd omdat deze niet past in de visie van het project. Soms worden ze geweigerd omdat iemand anders er toevallig ook al mee bezig is. Voor de maintainer, zeker van kleine projecten die toch al weinig contributors hebben, is het ook niet leuk om een nieuwe contributor op die manier weg te moeten sturen.

Stuur dus altijd even een berichtje. Dit kan via een comment in een issue, via het Gitter chatkanaal, via de mailinglist van het project, of via een andere weg die staat aangegeven in de README.md van het project, en wacht tot ze je een go geven. Meestal zijn maintainers heel blij als je dit doet, en zijn ze graag bereid om je vragen te beantwoorden.

Iets anders waar je op moet letten als je een feature bouwt, is dat je je houdt aan de coding standards van het project, ook als je het niet eens bent met die standards. Vaak worden deze afgedwongen in de build of beschreven op de website. Zo niet, kijk dan goed naar de bestaande code en zorg ervoor dat jouw code daar zo veel mogelijk op lijkt.

Tot slot geldt ook hier: volg de templates bij het indienen van je pull request. Heeft het project een CONTRIBUTING.md bestand? Volg dan ook de instructies in dit bestand. En onthoud: het openen van een PR is niet het einde. Je krijgt waarschijnlijk commentaar van de reviewers. Ga in gesprek en poets de code op tot het geaccepteerd wordt.

Gefeliciteerd! Je eerste PR is gemerged! Dit is je eerste stap op weg om vaste contributor, en misschien zelfs project maintainer te worden.

## Start je eigen project

Als je niet wil aanhaken bij iets wat al bestaat, maar een eigen project wil starten, kan dat natuurlijk. Maar hoe zorg je ervoor dat je

### WAAR MOET MIJN PROJECT OVER GAAN?

Gek genoeg zijn het vaak de saaie infrastructuurprojecten die actief zijn. JavaParser is bijvoorbeeld niet bepaald een spannend project. Het lag in een hoekje te verstoffen met een stapel bugreports die niemand wilde oplossen. Er was dus wel behoefte aan, maar niemand nam de verantwoordelijkheid om het te onderhouden. Danny heeft dit project met succes nieuw leven ingeblazen. Ook EqualsVerifier is niet sexy. Er bestond al een ander project met vergelijkbare functionaliteit: EqualsTester. Jan wilde het gebruiken, maar was het niet eens met de keuzes die de maker van dat project had genomen en besloot zijn eigen tool te bouwen. De truc is dus: kies een onderwerp waar mensen behoefte aan hebben. Inhoudelijke kennis van het onderwerp is niet eens nodig; die bouw je vanzelf op als je met het project aan de slag gaat.

**TRIAGE VAN OPEN ISSUES IS OOK EEN DANKBARE TAAK. LEES DE OPEN ISSUES EN VRAAG DE AANMELDERS OM MEER INFORMATIE WAAR NODIG**

project het niveau ontstijgt van knutselproject met maar 1 ster (je eigen) op GitHub? Een saaie maar belangrijke beslissing die je al vroeg moet maken is onder welke licentie je je project vrijgeeft. Over het algemeen geldt: hoe strikter de licentie, hoe minder gebruikers. Apache 2 is bijvoorbeeld een heel geschikte voor open source libraries. Er zijn tools die je kunnen helpen kiezen[4].
Ook is het handig om het jezelf zo makkelijk mogelijk te maken, door al in een vroeg stadium zo veel mogelijk te automatiseren. Zet een CI pipeline op in GitHub Actions of CircleCI, en zoek uit of je ook je releases vanuit CI kunt doen. Nu kun je gaan bouwen!

## Rollen

Zolang jij de enige maintainer van het project bent, vervul je meerdere rollen tegelijk: die van community manager en die van project maintainer. Veel projecten zullen dit stadium nooit ontgroeien. Dat betekent overigens niet dat dat het project niet succesvol kan zijn!
Er zijn honderden tools en libraries die veel gebruikt worden en maar één vaste maintainer hebben.
Voor de bus-factor[5] van je project is het wel goed om een vaste community om je project te bouwen. Wanneer dat lukt, kun je ook de rollen verdelen.

## Community manager

De meeste mensen denken direct aan techniek bij open source projecten, maar die techniek wordt gemaakt door mensen. Een community manager kan een aangename community cultiveren die mensen aantrekt, om zo te zorgen dat het project groeit.
Mensen staan erom bekend dat ze in allerlei soorten en maten komen. Een community manager let op deze menselijke kant door communicatie soepel te laten lopen en hier en daar hard in te grijpen. Niets zo vernietigend voor een community als ergernis en drama. Probeer dus altijd netjes en correct te blijven, zelfs als de andere persoon duidelijk een eikel is.
Nieuwe issues en PR's die ontvangen worden met ijzige stilte zorgen ervoor dat een project dood lijkt. Niemand gebruikt een project waar geen leven in zit. Een community manager moet daarom nieuwe issues en PR's in goede banen leiden. Een hulpmiddel hierbij zijn bijvoorbeeld templates die nieuwe contributors helpen alle benodigde informatie aan te leveren. Dit maakt het makkelijker om issues en PR's te beoordelen, maar maak ze niet te ingewikkeld: dan haken mensen af en mis je feedback. EqualsVerifier maakt hier veel gebruik van.
Daarnaast is er een grote hoeveelheid werk aan communicatie met andere projecten, met bedrijven, voor promotie, dat zonder community manager vaak links blijft liggen.

## Project Maintainer

Een maintainer houdt het stof van een project. De code zelf moet blijven compileren, de dependencies moeten niet verouderen, de buildstraat moet blijven draaien, documentatie moet up-to-date blijven, releases moeten blijven verschijnen. Dit demonstreert aan de buitenwereld dat het project actief is, en actieve projecten trekken gebruikers aan. Trek je gebruikers aan, dan krijgt een project feedback en blijft het actief. Een vicieuze cirkel.
Elke wijziging aan de code moet binnen redelijke tijd in een nieuwe release terechtkomen - hier is "release early, release often" een goede hulp. Het JavaParser project maakt bijvoorbeeld in principe elke week een nieuwe release. Als project maintainer ben je ook verantwoordelijk voor de planning van je project. Bepaal waar je naartoe wil en houd dit in de gaten. De meeste nieuwe contributors fixen namelijk hun eigen ding en werken niet mee aan het grote plaatje. Voorkom dat je project "a patchy server" wordt!

## Conclusie

Open source kan alleen bestaan door mensen als jij en ik. De tijd en moeite die je erin steekt geeft je waardering van anderen, maar ook waardering voor het werk van anderen. En het geeft een heel goed gevoel wanneer bijvoorbeeld iemand die je niet kent, naar je toe komt en zegt hoe graag ze je project gebruiken.

Wij wensen je veel plezier met je eerste stappen! ■

**ALS PROJECT MAINTAINER BEN JE OOK VERANTWOORDELIJK VOOR DE PLANNING VAN JE PROJECT. BEPAAL WAAR JE NAARTOE WIL EN HOUD DIT IN DE GATEN**

## REFERENTIES

[1] **https://javaparser.org**
[2] **https://jqno.nl/equalsverifier**
[3] **http://up-for-grabs.net**
[4] **https://choosealicense.com**
[5] **https://en.wikipedia.org/wiki/Bus_factor**