

# Geen webservice? Geen probleem!

Webservice mocking met WireMock

Bas Dijkstra

[www.ontestautomation.com](http://www.ontestautomation.com)

[bas@ontestautomation.com](mailto:bas@ontestautomation.com)

[@\\_basdijkstra](https://twitter.com/_basdijkstra)

WiFi

# Wat gaan we doen?

\_Stubbing, mocking en servicevirtualisatie

\_WireMock

\_Aan de slag

# Vorbereitung

\_ Installatie Eclipse (of een andere IDE)

\_ Installatie TestNG-plugin (voor Eclipse)

\_ Installatie m2e (of tegenhanger voor andere IDE)

\_ Importeren Maven-project in IDE

\_ Update project (Eclipse) of tegenhanger

# Problemen met testomgevingen

- \_ Systemen bestaan uit veel componenten

- \_ Onbeschikbaarheid voor testdoeleinden

- \_ Parallel ontwikkelen

- \_ Geen controle over testdata

- \_ Kosten voor gebruik third party component

- \_ ...

# Simulatie tijdens testuitvoer

- \_ Simuleren van gedrag van afhankelijkheden

- \_ Onbeschikbaarheid voor testdoeleinden

  - \_ Parallel ontwikkelen

  - \_ Geen controle over testdata

  - \_ Kosten voor gebruik third party component

  - \_ ...

# Stubbing

\_Vooraf gedefinieerde responses

\_Geen flexibiliteit

\_Statusverificatie

# Mocking

\_ Gedrag kan worden gedefinieerd tijdens initialisatie test

\_ (Iets) meer flexibiliteit

\_ Verificatie van gedrag



# Servicevirtualisatie

\_ Simuleren van gedrag van complexe afhankelijkheden

\_ 'Next level' stubbing / mocking

\_ Ondersteuning veelheid aan protocollen

\_ Data driven

# WireMock

`_http://wiremock.org`

`_Java`

`_HTTP mock server`

`_ werkt dus alleen voor HTTP(S)`

`_open source`

`_ontwikkeld en onderhouden door Tom Akehurst`

# WireMock configureren

\_JAR file toevoegen aan Java-project

\_Maven (invoegen screenshot met dependency)

\_Gradle (invoegen screenshot)

# Een voorbeeldmock

\_Invoegen eenvoudig voorbeeld waarin veelgebruikte opties worden getoond

# Handige WireMock features

\_Invoegen aantal features

\_Volledige documentatie op  
<http://wiremock.org/docs/>

# WireMock standalone draaien

\_ Starten WireMock server

\_ \_Opties: port, keystore, ...

\_ Mocks permanent beschikbaar stellen voor meerdere teams

\_ Mocks herconfigureren via JSON

\_ Invoegen voorbeeldcommando

# WireMock starten en stoppen tijdens testuitvoer

\_Integratie in testuitvoer

\_Meeleveren mocks met testsuite in versiebeheer

\_Integratie met JUnit (invoegen voorbeeld)

\_Kan ook zonder gebruik te maken van JUnit

# Demo

Gebruik van WireMock in tests



# Aan de slag!

`_WireMockExercises1`

`_Opzetten eenvoudige mocks`

`_Opdrachten staan in commentaar`

`_Controleren van werking door uitvoeren tests`

# Record & playback-opties

\_Gebruik WireMock als proxy

\_Opnemen request-response pairs (traffic)

\_Genereren mock uit deze traffic

# Record & playback-opties

\_Gebruik WireMock als proxy

\_Opnemen request-response pairs (traffic)

\_Genereren mock uit deze traffic

# Demo

Gebruik van record & playback in WireMock

# Voor- en nadelen van record & playback

## \_Voordelen:

- \_ Snel opzetten mocks
- \_ Analyseren traffic met onbekende specificaties

## \_Nadelen:

- \_ Bij wijziging alles opnieuw opnemen
- \_ Mocks zijn niet flexibel
- \_ Mocks zijn slecht uitbreidbaar

\_Vergelijk met record & playback bij testautomatisering

# Request matching

\_Bepalen welk antwoord wordt teruggestuurd op basis van specifieke eigenschappen request

\_Opties:

\_ URL

\_ HTTP-methode

\_ Query-parameters

\_ Headers

\_ Elementen in body van request

\_ ...

Voorbeeld: matching op URL

`_Invoegen voorbeeld`

Voorbeeld: matching op body-  
element

`_Invoegen voorbeeld`



Voorbeeld: matching op body-  
element

`_Invoegen voorbeeld`

# Aan de slag!

\_WireMockExercises2

\_Gebruik request matching

\_Opdrachten staan in commentaar

\_Controleren van werking door uitvoeren tests

# Foutsimulatie

\_ Uitbreiden van testdekking door simulatie fouten

\_ Vaak lastig te doen in 'echte' systemen

\_ Eenvoudig met mocks en stubs

\_ Testen foutafhandeling testobject

Voorbeeld: foutsimulatie door  
HTTP-statuscode

\_Invoegen voorbeeld

Voorbeeld: foutsimulatie door  
timeout

\_Invoegen voorbeeld

Voorbeeld: foutsimulatie door  
bad response

\_Invoegen voorbeeld

# Aan de slag!

`_WireMockExercises3`

`_Gebruik foutsimulatie`

`_Opdrachten staan in commentaar`

`_Controleren van werking door uitvoeren tests`

# Stateful mocks

\_ Mocks tot nu toe waren stateless

\_ Volgorde van aanroepen methoden maakte niet uit

\_ Dit is in de 'echte' wereld niet altijd het geval

\_ Request A > request B geeft ander gedrag dan  
request B > request A



# Stateful mocks: een voorbeeld

\_ Winkel met 2 exemplaren van een artikel in  
\_ voorraad

\_ Eerste call naar addToCart: OK

\_ Tweede call naar addToCart: OK

\_ Derde call naar addToCart: NOK (foutmelding,  
\_ foutcode, ...)

# Stateful mocks: een voorbeeld

`_Invoegen implementatie voorbeeld`

# Aan de slag!

`_WireMockExercises4`

`_Gebruik stateful mocks`

`_Opdrachten staan in commentaar`

`_Controleren van werking door uitvoeren tests`

# Andere nuttige features

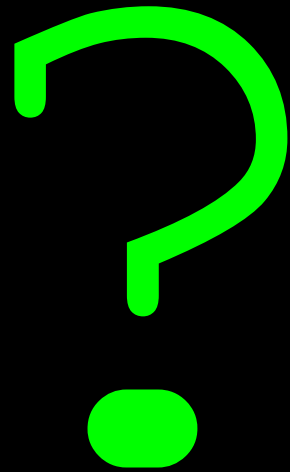
\_ Verificatie (is een bericht bij de mock  
aangekomen?)

\_ Transformatie van responses (via extensions)

\_ Integratie in een CI / CD pipeline

\_ Documentatie: <http://wiremock.org/docs/>

Vragen



# Contact

\_Email: [bas@ontestautomation.com](mailto:bas@ontestautomation.com)

\_Weblog: <http://www.ontestautomation.com>

\_Twitter: [@\\_basdijkstra](https://twitter.com/_basdijkstra)