****

Testplan template

Integration and Communication

CODE TCIF-V2IAC1-15

HBO-ICT SIE Jaar 2 2016-2017

Cursuseigenaar: jeroen.weber@hu.nl

Auteurs

Alex Jongman

Jeroen Weber

**Review**

BAR:

Nick Stuivenberg, 1667775

Datum

20 – 06 – 2017

Versie 0.2

© Hogeschool Utrecht, Utrecht, 2015



# Inhoudsopgave

[Inhoudsopgave 2](#_Toc473464760)

[1. Versie 3](#_Toc473464761)

[2. Introductie 3](#_Toc473464762)

[3. Testomgeving 3](#_Toc473464763)

[4. Smoketest 3](#_Toc473464764)

[5. Testcases black box test (integratietest of functionele test) 3](#_Toc473464765)

[6. Unit Tests 3](#_Toc473464766)

[7. Regressietests 4](#_Toc473464767)

# Versie

|  |  |  |
| --- | --- | --- |
| Datum | Versie | Aanpassing |
| 19-06-2017 | 0.1 | Document aangemaakt, h1, h3, h6, h7 |
| 20-06-2017 | 0.2 | H4 |

# Introductie

Deze is getest met een unit test, smoke test en via rest calls via Soapui.

# Testomgeving

Er zijn twee verschillende soorten testen voor deze opdracht geschreven. Dat zijn de JUNIT testen en de integratietesten.

De Junit (versie 3) worden automatisch gelopen wanneer er een mvn clean install wordt gedaan. Mocht een van de testen falen dan volgt er een error.

Daarnaast is er een Integratie test geschreven die een SOAP bericht stuurt. Het is een integratietest, dus dat houdt in dat deze alleen werkt wanneer de hele server runt. In dit geval wordt er een createHouse Request in een SOAP envelop verstuurd en de response van de server wordt afgedrukt.

# Smoketest

|  |  |
| --- | --- |
| Gebruikers voert niet Nederlandse postcode in | Systeem geeft error |
| Gebruiker voert een string in bij huisnummer | Systeem geeft error |
| Gebruiker voert een plaatsnaam in met 40 tekens | Het systeem weet dat deze niet bestaan in Nederland, error |
| Gebruiker doet alsof hij in een paleis woont en veertig kamers heeft | Max is 20. Systeem geeft errror. |
| Gebruiker heeft als achternaam: “van den Heuvel tot Beichlingen, gezegd Bartolotti Rijnders” (aangepast) | Systeem geeft onterecht error. Dit is een voorkomende achternaam en dient geaccepteerd te kunnen worden. |

# Testcases black box test (integratietest of functionele test)

Tijdens het bouwen van deze opdracht is er niet op beveiliging gelet. Er zijn vier Requests en Vier responses gemaakt. Deze zijn Create, Read, Update en Delete.

De Create,Read en Update response geven het volledige object terug. De Delete response geeft alleen een bericht terug indien gelukt.

Create ontvangt een volledig object request, Read, Update en Delete Hebben Straatnaam, huisnummer, huisnummertoevoeging en postcode nodig. Op deze manier weten we wanneer het over een uniek huis gaat.

Tijdens het createn, readen en updaten wordt gekeken of het huis al bestaat. Bij createn wordt ern een response ‘DUBBEL!’ teruggegeven indien deze fout is.

Deleten geeft een response terug als het verwijderen gelukt is.

# Unit Tests

package test;

import junit.framework.TestCase;

import nl.hu.fnt.gsos.domain.House;

import nl.hu.fnt.gsos.wsinterface.CreateHouseResponse;

import nl.hu.fnt.gsos.wsinterface.UpdateHouseResponse;

// -- Junit 3! Niet 4

public class MyFirstUnitTest extends TestCase {

public void testcreateTestHouseNaamEigenaar() {

CreateHouseResponse response = new CreateHouseResponse();

response.setStraatnaam("Laan");

response.setHuisnummer(13);

response.setHuisnummertoevoeging("c");

response.setPostcode("3404GV");

response.setStad("zeist");

response.setAantalkamers(1);

response.setAchternaameigenaar("kroes");

House h = new House();

h.setStraatnaam(response.getStraatnaam());

h.setHuisnummer(response.getHuisnummer());

h.setHuisnummertoevoeging(response.getHuisnummertoevoeging());

h.setPostcode(response.getPostcode());

h.setStad(response.getStad());

h.setAantalkamers(response.getAantalkamers());

h.setAchternaameigenaar(response.getAchternaameigenaar());

assertEquals(response.getAchternaameigenaar(), h.getAchternaameigenaar());

}

public void testUpdateHouseEigenaar() {

UpdateHouseResponse response = new UpdateHouseResponse();

House h = new House();

h.setStraatnaam("Laan");

h.setHuisnummer(13);

h.setHuisnummertoevoeging("c");

h.setPostcode("3404GV");

h.setStad("zeist");

h.setAantalkamers(1);

h.setAchternaameigenaar("kroes");

// -- stiekem alles veranderen

response.setStraatnaam("Weg");

response.setHuisnummer(132);

response.setHuisnummertoevoeging("cd");

response.setPostcode("3434GV");

response.setStad("ueist");

response.setAantalkamers(2);

response.setAchternaameigenaar("kloes");

h.setStraatnaam(response.getStraatnaam());

h.setHuisnummer(response.getHuisnummer());

h.setHuisnummertoevoeging(response.getHuisnummertoevoeging());

h.setPostcode(response.getPostcode());

h.setStad(response.getStad());

h.setAantalkamers(response.getAantalkamers());

h.setAchternaameigenaar(response.getAchternaameigenaar());

assertEquals(response.getAchternaameigenaar(), h.getAchternaameigenaar());

}

public void testUpdateHouseUitbreiding() {

UpdateHouseResponse response = new UpdateHouseResponse();

House h = new House();

h.setStraatnaam("Laan");

h.setHuisnummer(13);

h.setHuisnummertoevoeging("c");

h.setPostcode("3404GV");

h.setStad("zeist");

h.setAantalkamers(1);

h.setAchternaameigenaar("kroes");

response.setStraatnaam("Laan");

response.setHuisnummer(13);

response.setHuisnummertoevoeging("c");

response.setPostcode("3404GV");

response.setStad("zeist");

response.setAantalkamers(6);

response.setAchternaameigenaar("kroes");

h.setStraatnaam(response.getStraatnaam());

h.setHuisnummer(response.getHuisnummer());

h.setHuisnummertoevoeging(response.getHuisnummertoevoeging());

h.setPostcode(response.getPostcode());

h.setStad(response.getStad());

h.setAantalkamers(response.getAantalkamers());

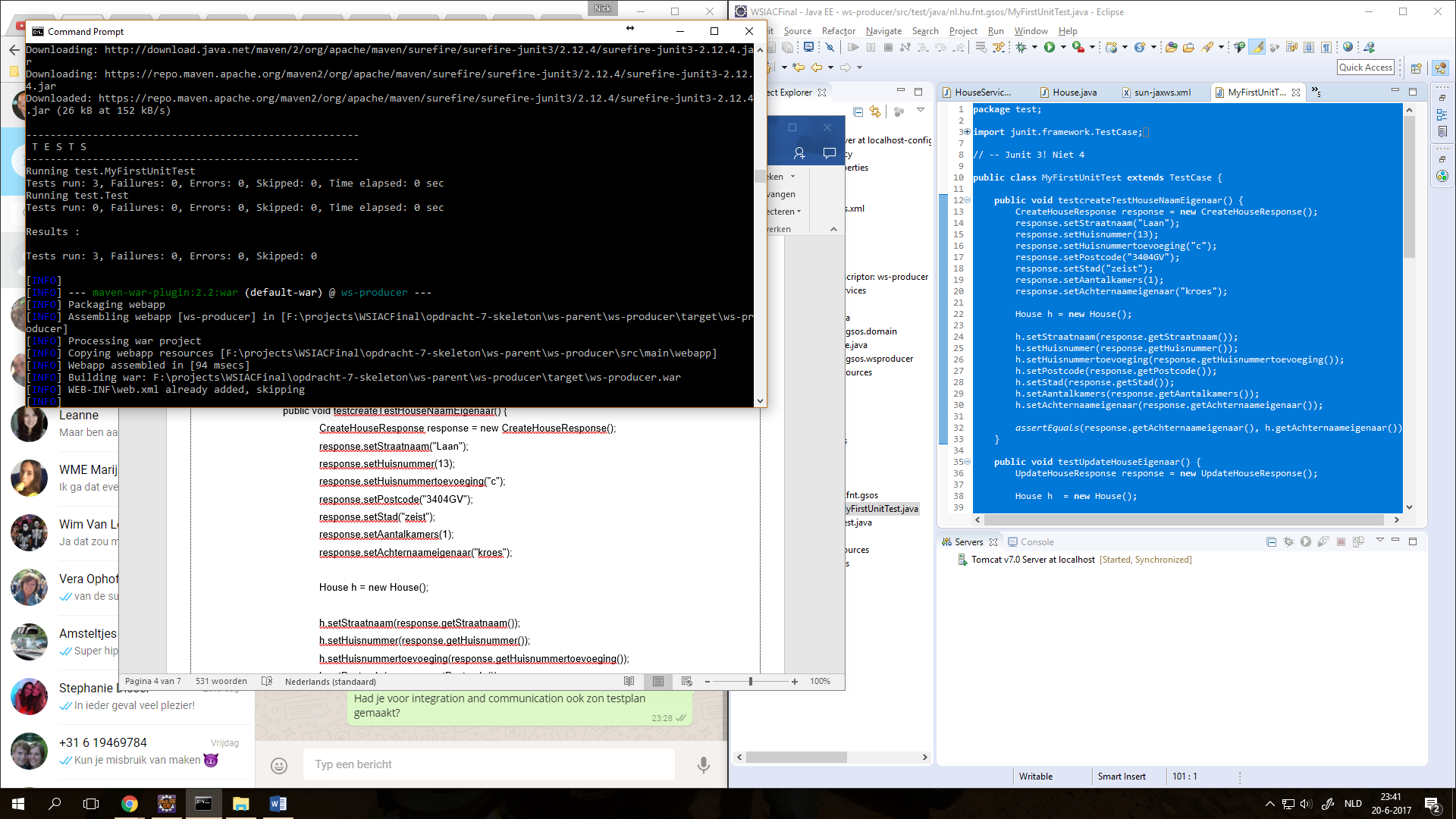
h.setAchternaameigenaar(response.getAchternaameigenaar());

assertEquals(response.getAantalkamers(), h.getAantalkamers());

}

}

Ik heb hier een CRUD applicatie gemaakt met als opdracht om het simpel te houden. Dat heb ik dan ook gedaan. Daaruit volgend krijg je alleen maar ‘simpele’ methode en ‘domme’ tests.



# Regressietests

Maven doet dit tijdens het builden. Daarnaast kan er gekozen worden om gebruik te maken van bijvoorbeeld Selenium testen om de calls te testen. Mocht er gebruikt gemaakt worden van een javascript front end die deze SOAP messages moet verwerken (kan me niet voorstellen), dan kan dit getest worden met Jasmine voor (automatische) Use case testing en Karma indien Angular wordt gebruikt.