Testplan IAC

Dion de Koning

V2C

[**Introductie**](#_fngqhgbigy93) **3**

[**Testomgeving**](#_kju7ofyf23f1) **4**

[**Smoke test**](#_6kzn0su9xdz8) **5**

[**Testcases black box test**](#_7lbflya4t4ja) **6**

[**Unit Tests**](#_f0zx28bft6zu) **7**

[**Regressietest**](#_mrictqhaiqpl) **8**

# Introductie

Voor deze opdracht moest ik een SOAP webservice schrijven waarmee je je BMI kon berekenen. Deze service geeft vervolgens je BMI en een advies terug. Om dit zo goed mogelijk te testen zal ik eerst een smoketest uitvoeren. Hiermee kun je testen of de service daadwerkelijk werkt.

Vervolgens zal ik een black box test uitvoeren. Hiermee zal ik testen of bij bepaalde input ook de juiste output eruit komt. Dit zal gebeuren op basis van de WSDL.

Verder zal ik ook unit-en regressietests uitvoeren. Hiermee controleer je of je code ook daadwerkelijk werkt.

# 

# Testomgeving

In dit testdocument ga ik mijn BMICalculator testen. Deze service kan een BMI op basis van je gewicht en lengte in centimeters berekenen. Ik ga de service testen door middel van JUnit en SoapUI. Als je met maven de nieuwste versie van de service build wordt er automatisch een JUnit test uitgevoerd. Deze controleert of de bestaande functionaliteit nog werkt. Wanneer deze slaagt wordt er een root.war gegenereerd. Deze is vervolgens te implementeren op een productieserver. Als de service draait laat ik SoapUI een test suite runnen. Deze stuurt verschillende requests naar de service. Hiervan test de eerste alleen of de service te bereiken is. Daarna wordt de response en de foutafhandeling getest. Deze moeten een verwacht resultaat teruggeven. Gebeurt dit niet dan faalt de test.

# 

# Smoke test

Om de werking van het systeem te testen stuur SoapUI een leeg xml request. Wanneer deze antwoord krijgt is de smoke test geslaagd. Dan geeft de server namelijk antwoord dus werkt deze. Wanneer de smoke test niet slaagt is de service niet beschikbaar. Vervolgens wordt er ook nog een ongeldig bericht gestuurd. Deze bevat een waarde waardoor er in de service door 0 gedeeld moet worden. Dit zou dan een foutmelding terug moeten sturen. Als deze tests slagen is de smoke test geslaagd.

# 

# Testcases black box test

Voor de black box tests heb ik SoapUI gebruikt. Deze voert de volgende stappen uit:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stap | Gewicht | Lengte | Uitkomst | Voldaan |
| Smoketest | geen | geen | response object | n.v.t. |
| Normal test 1 | 64 | 164 | 23.8 | n.v.t. |
| Fault test | 0 | 64 | soap fault | n.v.t. |
| Normal test 2 | 60 | 170 | 20.8 | n.v.t. |

Wanneer bovenstaande test slagen is de black box test succesvol. Vervolgens kan er vanuit gegaan worden dat de applicatielogica naar behoren werkt.

# 

# Unit Tests

De unit tests worden uitgevoerd door JUnit. Dit framework voert een test uit in de buildcycle van maven. Als de tests niet slagen stopt maven met compilen. Omdat het een vrij simpele webservice is is de unittest niet heel uitgebreid. Met de volgende code wordt getest of de functionaliteit van de applicatie nog steeds correct is:

public class BMICalcTest {

private static BMICalc *calc*;

@BeforeClass

public static void initCalc(){

*calc* = new BMICalc(164, 64);

}

@Before

public void beforeEachTest(){

System.*out*.println("Voor elke test");

}

@After

public void afterEachTest(){

System.*out*.println("Na elke test");

}

@Test

public void testCalculate(){

String result = *calc*.calculate();

*assertEquals*("23.8" , result);

}

}

Daarbij is ook de dependency aan maven toegevoegd:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

Elke keer als maven gaat builden wordt de bovenstaande test uitgevoerd. Hiermee wordt vervolgens gekeken of de basis nog steeds werkt.

# Regressietest

Voor de regressietest heb ik JUnit toegevoegd. Hiermee wordt voordat de applicatie met maven gebuild wordt altijd getest of de vorige functionaliteit nog werkt. Op deze manier zit er altijd nog een check tussen.