Testrapport   
IAC Herkansingsopdracht 2017  
Smart Cities SOAP Service

“Gemiddelde berekenen”

Jerre van den Berg

1649996

Contents

# Smoketest

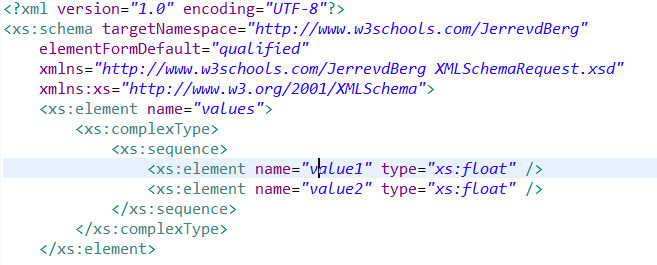
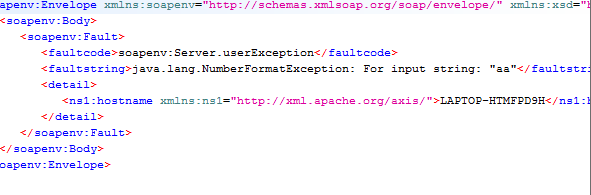
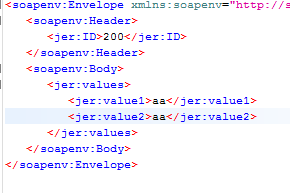
  
Om het system te checken of het hufterproof is, en of de onderliggende XSD files goed zijn opgesteld, wordt met behulp van SoapUI een stringwaarde in het request geplaatst.  
  
In de XSD’s wordt dit afgevangen, door de restrictie dat het request alleen maar een float mag bevatten(zie figure 1).   
  
  
  
  
  
  
  
Hieronder het voorbeeld wanneer in de request een string wordt ingevuld(zie figure 2&3)

Figure 3

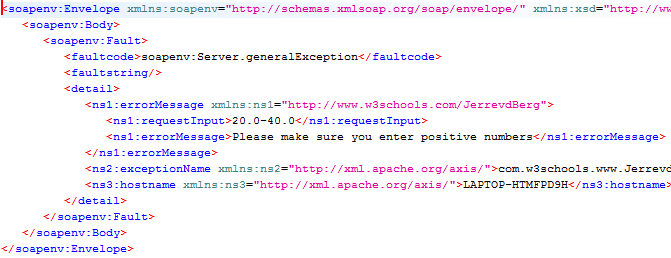
Figure

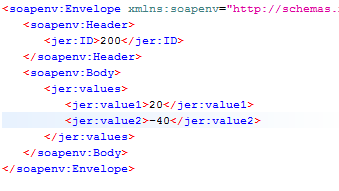


Figure

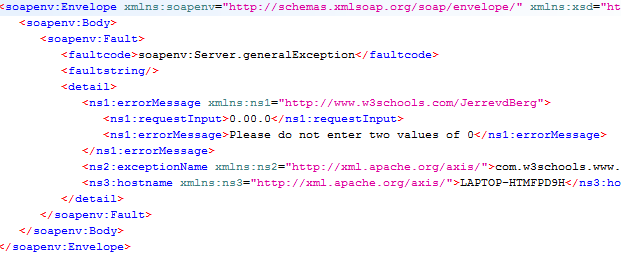
Geheel in de verwachting wordt er een exception gegooid omdat de request niet in het juiste formaat is. Hiermee kan geconcludeerd worden dat de XSD de input goed afvangt.

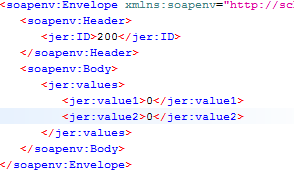
# Functionele test

In dit hoofdstuk zal getest worden of de SOAP-service ook daadwerkelijk het juiste gedrag vertoond.  
Het doel van de service is om het gemiddelde te berekenen van twee waardes. Deze waardes mogen niet negatief zijn en ook mogen beide waardes niet 0 zijn.  
  
In de volgende screenshots wordt met SoapUI getest of bij het overschrijden van de voorwaarden (niet negatief en niet beide 0) er een (goede) errormessage wordt getoond en er wordt getest of de berekening juist wordt gemaakt.  
  
Case 1: Invoeren van een negatief getal



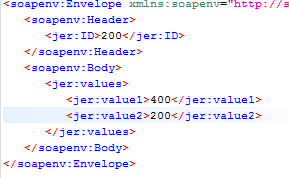
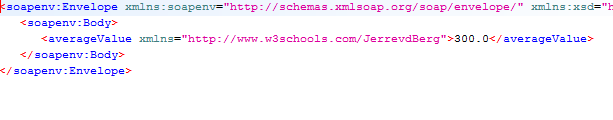
Zoals te zien is in de bovenstaande figuren, wordt er bij het invullen van een negatief getal, een errormessage getoond die aangeeft dat de gebruiker geen negatieve getallen mag invoeren.  
  
Case 2: Invoeren van twee 0-waarden





Zoals te zien is, wordt het invoeren van twee nul waardes correct afgevangen door de service. De gebruiker krijgt een duidelijk foutmelding.

Case 3: Invoeren van twee floats en de verwachtte reactive



We zien hier dat er twee juiste waardes ingevuld worden, namlijk 400 en 200. Het gemiddelde hiervan is 600/2 = 300. Het resultaat is dus juist en de berekening is juist gemaakt.  
  
Ook zien we dat de juiste methode wordt aangeroepen, namelijk averageValue. Dit is ook zo gedefinieerd in de WSDL. Het resultaat is tevens in float formaat, zoals gespecifieerd in de XSD.  
  
In al deze cases bevatte de requests ook de gespecificeerde header.  
  
De blackbox test is dus succesvol gelukt.

# Unit Tests

Hierbij de geschreven Unit tests in Eclipse (java)

**package** test.java;

**import** com.w3schools.www.JerrevdBerg.Values;

**import** junit.framework.\*;

**public** **class** TestValues **extends** TestCase {

**public** **void** testSetValue1(){

Values v1 = **new** Values(1.0f, 50.0f);

v1.setValue1(20.0f);

*assertEquals*(v1.getValue1(), 20.0f);

}

**public** **void** testSetValue2(){

Values v2 = **new** Values(1.0f, 50.0f);

v2.setValue1(40.0f);

*assertEquals*(v2.getValue1(), 40.0f);

}

**public** **void** testGetValue1(){

Values v1 = **new** Values(1.0f, 50.0f);

*assertEquals*(v1.getValue1(), 1.0f);

}

**public** **void** testGetValue2(){

Values v2 = **new** Values(1.0f, 300.0f);

*assertEquals*(v2.getValue2(), 300.0f);

}

}

De tests runnen succesful in zowel maven (mvn test) als JUnit debug (Eclipse).