|  |
| --- |
| Hogeschool Utrecht |
| Testplan |
| IAC |
| Roy Hendriks [Klas: SIE-V2A] |

|  |  |  |
| --- | --- | --- |
| **Testplan** | | |
| **Document datum** | 23-06-2017 | |
| **Document versie** | V1.0 | |
| **Versiebeheer** | V1.0 | Geen Verandering |
|  |  |  |

Inhoudsopgave

[Introductie 2](#_Toc477106417)

[Testomgeving 2](#_Toc477106418)

[Smoketest 3](#_Toc477106419)

[Testcases black box test 4](#_Toc477106420)

[Unit test 6](#_Toc477106421)

# Introductie

In deze opdracht ga ik een SOAP web service realiseren en testen. De service moet van toepassing zijn binnen het domein “Smart Cities”. Ik houd de service simpel en bedenk dat ik met berekeningen bijna elk domein afdek. De volgende eindproducten maak ik voor de door mij gekozen service:

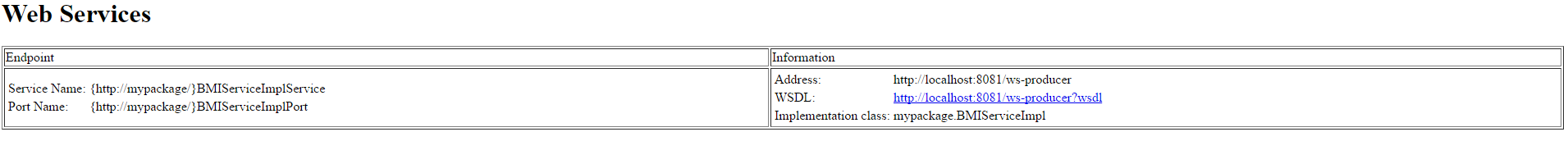
* XML Schema (XSD)
* WSDL
* SOAP
* Java Webservice
* Testplan
* Testrapport

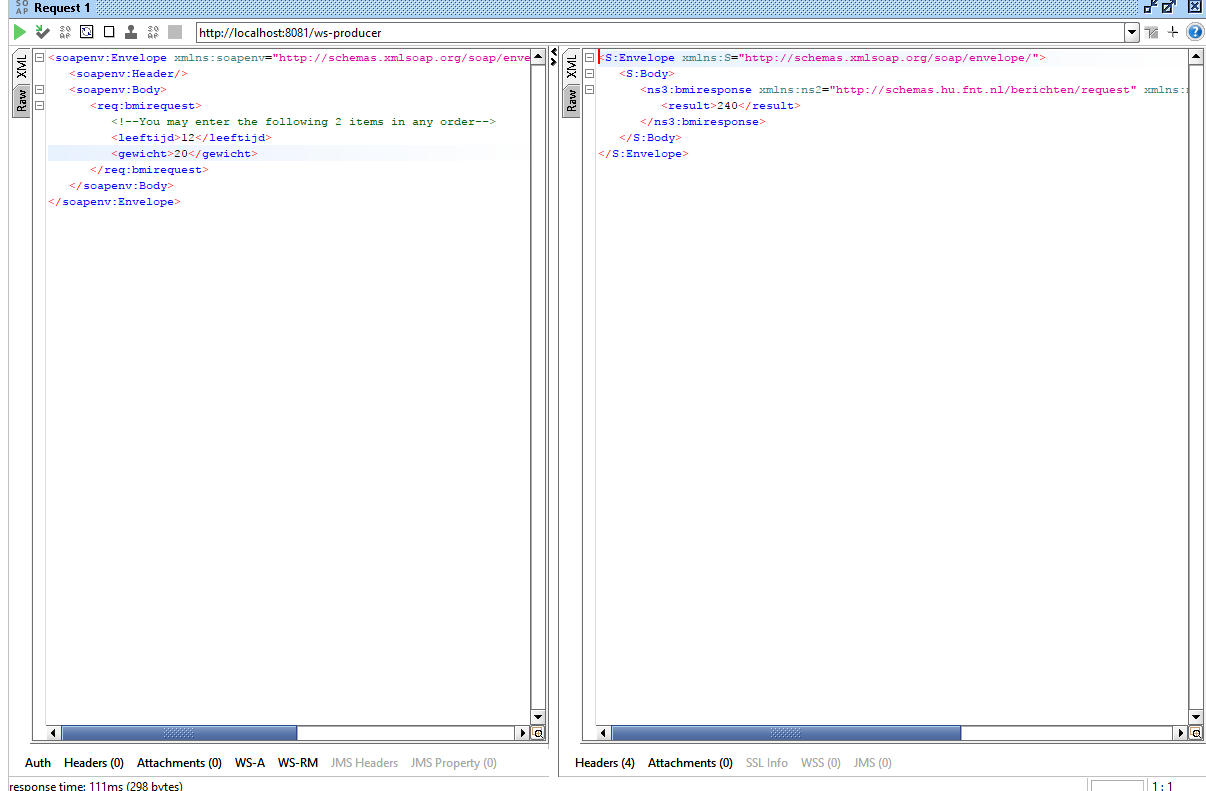
De service die ik moet maken is een service die de BMI berekend. De formule van BMI klopt niet maar ik laat zien dat de SOAP service werkt. De tests die ik voor deze service ga uitvoeren zijn de: smoketest, functionele test, unit tests, regressietests.

# Testomgeving

Ik ga mijn service testen door middel van SoapUI en JUnit. Met JUnit kan ik de berekening testen en met SoapUI kan ik het response testen.

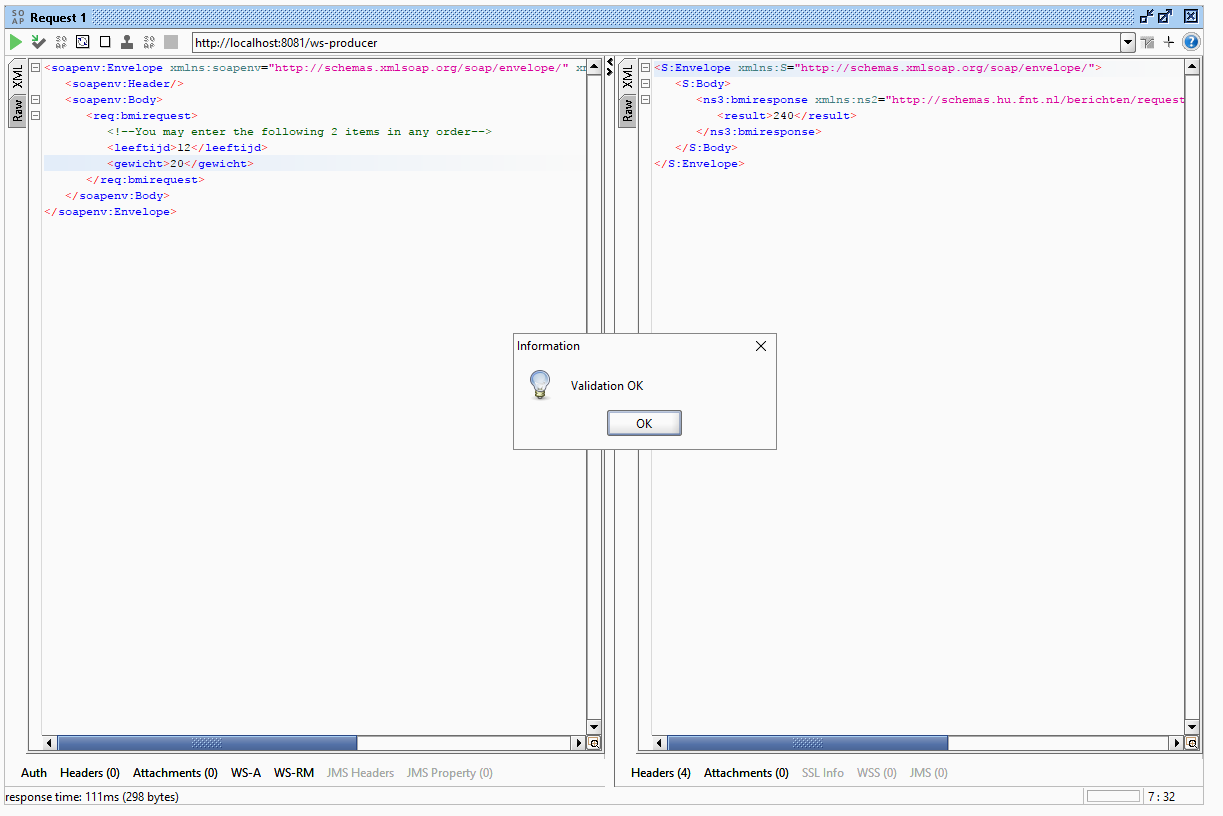
# Smoketest



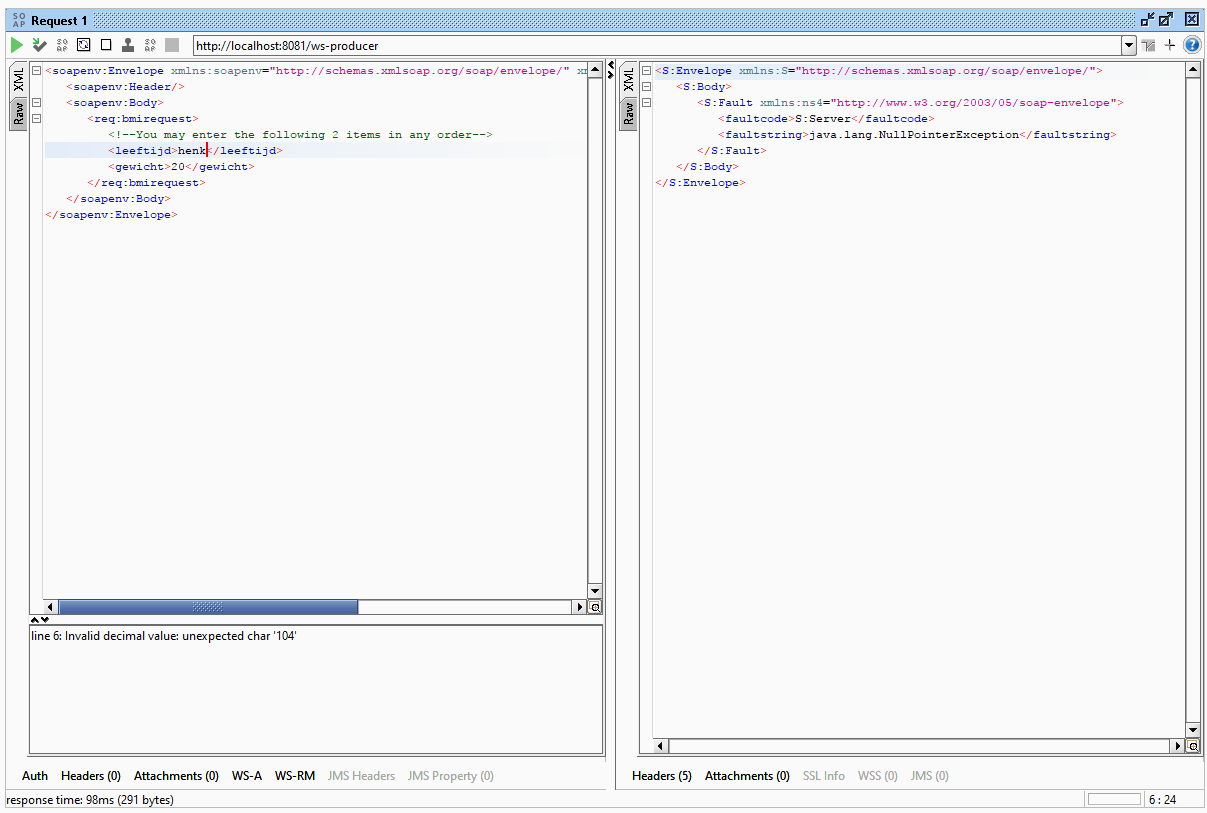


# Testcases black box test

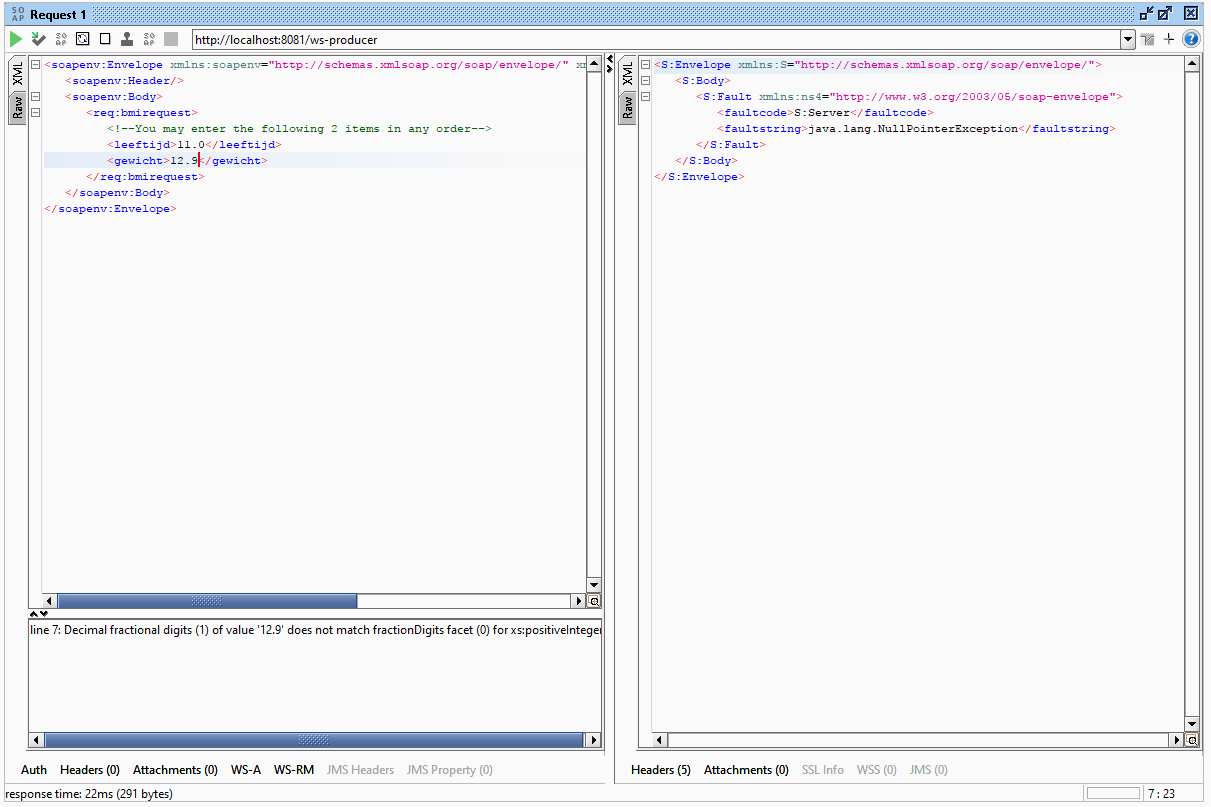
Bij invulling van een geheel positief getal moet er een goed resultaat komen:



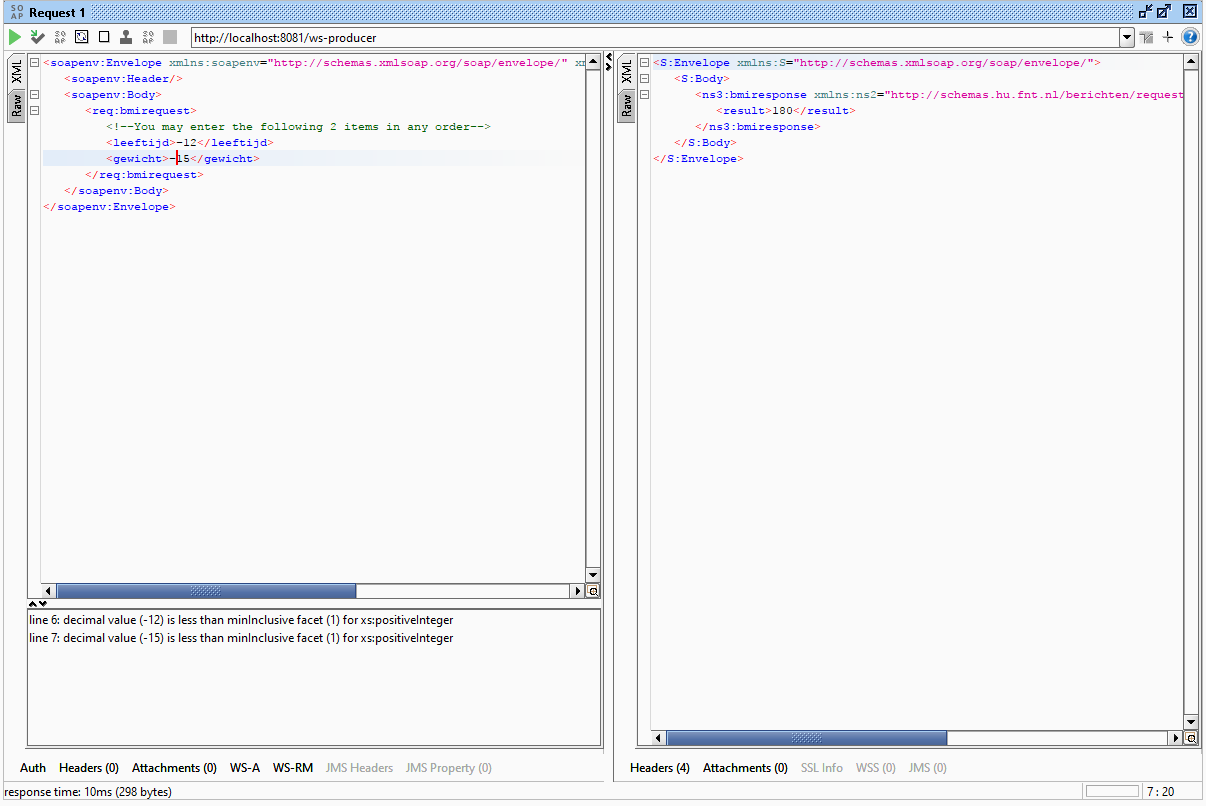
Bij invulling van een string moet er een error komen dat het geen string mag zijn:



Bij invulling van een double moet er een error komen dat er geen double gegeven mag worden:



Bij invulling van een negatief getal moet er een error komen dat er geen negatief getal gegeven mag worden:



# Unit test

De unit test is gemaakt in JUnit.

@WebService( endpointInterface= "mypackage.WSInterface")  
public class BMIServiceImpl implements WSInterface {  
 @Override  
 public Response calculatePower(Request request) {  
 ObjectFactory factory = new ObjectFactory();  
 Response response = factory.createResponse();  
 Double getal = request.getGewicht().doubleValue() \* (Math.*pow*(request.getLeeftijd().doubleValue(),2));  
 BigInteger actualResult = BigInteger.*valueOf*(getal.longValue());  
 response.setResult(actualResult);  
 return response;  
 }  
}

public class BMIServiceImplTest {  
 @Test  
 public void geldigeInput() {  
 Request req = new Request();  
 req.setGewicht(BigInteger.*valueOf*(50));  
 req.setLeeftijd(BigInteger.*valueOf*(25));  
 BMIServiceImpl wsi = new BMIServiceImpl();  
 Response result = wsi.calculatePower(req);  
 BigInteger result2 = result.getResult();  
 int actualResult = result2.intValue();  
 *assertEquals*(31250, actualResult);  
 }  
}

