Testplan en Testrapport Integration and Communication CODE TCIF-V2IAC1-15

HBO-ICT SIE Jaar 2 2016-2017

Cursuseigenaar: jeroen.weber@hu.nl

Auteurs
Alex Jongman
Jeroen Weber

Student
Naam: Kachun Tang
Studentnummer:1620293

Datum
23 – 06 – 2017

Versie 1.0

Inhoudsopgave

Inhoudsopgave **2**

1. Introductie **3**

2. Testomgeving **3**

3. Smoketest **4**

4. Functionele test **4**

5. Unit Tests **5**

6. Integratietest **6**

7. Regressietests **6**

1. Introductie

Het project omvat een webservice die de standaarddeviatie van een lijst cijfers kan berekenen. Als uitbreiding hierop kan de webservice ook de variantie en het gemiddelde berekenen. Daarnaast wordt het resultaat van de laatst uitgevoerde berekening bijgehouden. De hieronder beschreven testen zijn zowel uitgevoerd op de SOAP service als de REST service.

2. Testomgeving

De omgeving die getest gaat worden is een Java Web Applicatie die op Tomcat 7.0.78 draait. Hieronder staat een beschrijving van de services die deze applicatie biedt:

calculateMean	Berekend het gemiddelde van een lijst cijfers.
calculateVariance	Berekend de variantie van een lijst cijfers.
calculateStandardDeviation	Berekend de standaarddeviatie van een lijst cijfers.
getLastResult	Geeft de uitkomst terug van de laatst uitgevoerde berekening.

De volgende testen zullen uitgevoerd worden:

- Smoketest
- Functionele test
- Unit tests
- Integratietest
- Regressietests

Hiervoor zullen de volgende tools gebruikt worden:

- Chrome
- SoapUI
- JUnit

3. Smoketest

Om te testen of het systeem functioneert is ervoor gekozen om de wsdl pagina te testen. De URL hiervooor is: http://localhost:8080/statistics-soap-service/soap?wsdl

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<!--
    Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.8 svn-revision#13980.
 -->
<!--
    Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.8 svn-revision#13980.
 -->
<definitions targetNamespace="http://ws.iac.hu.nl/" name="StatisticsServiceImplService">
  <import namespace="http://nl.hu.iac/statistics-service/wsdl/" location="http://localhost:8080/statistics-soap-service/soap?wsdl=1"/>
  <binding name="StatisticsServiceImplPortBinding" type="ns1:statisticsServiceInterface">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="calculateStandardDeviation">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
      <fault name="Fault">
        <soap:fault name="Fault" use="literal"/>
      </fault>
```

4. Functionele test

Hieronder staan de operaties met input en output. De laatste operatie is afhankelijk van de voorafgaande.

Nr.	Operatie	Input	Verwachte output	
1	calculateMean	1.0, 2.0, 3.0, 4.0, 5.0	3.0	☑
2	calculateVariance	1.0, 2.0, 3.0, 4.0, 5.0	1.581	☑
3	calculateStandardDeviation	NaN	Fault	☑
4	calculateStandardDeviation	1.0, 2.0, 3.0, 4.0, 5.0	2.5	☑
5	getLastResult	/	2.5	☑

De testen zijn handmatig met SOAP UI uitgevoerd. Deze zijn op de volgende locatie terug te vinden: *src\main\webapp\WEB-INF\soap-messages*

5. Unit Tests

Er is gekozen voor het framework “JUnit 4” om de testen in te ontwikkelen.

Nr.1

```
@Test
public void calculateMean() throws Exception {
    StatisticsServiceImpl service = new StatisticsServiceImpl();

    Request request = createRequest(doubleList);

    Response response = null;
    try {
        response = service.calculateMean(request);
    } catch (Fault e) {
        e.printStackTrace();
        fail();
    }
    assertEquals(expectedMean, response.getResult(), 0.001);
}
```

Nr. 2

```
@Test
public void calculateVariance() throws Exception {
    StatisticsServiceImpl service = new StatisticsServiceImpl();

    Request request = createRequest(doubleList);

    Response response = null;
    try {
        response = service.calculateVariance(request);
    } catch (Fault e) {
        e.printStackTrace();
        fail();
    }
    assertEquals(expectedVariance, response.getResult(), 0.001);
}
```

Nr. 3

```
@Test
public void calculateStandardDeviation() throws Exception {
    StatisticsServiceImpl service = new StatisticsServiceImpl();

    Request request = createRequest(doubleList);

    Response response = null;
    try {
        response = service.calculateStandardDeviation(request);
    } catch (Fault e) {
        e.printStackTrace();
        fail();
    }
```

```
    assertEquals(expectedStandardDeviation, response.getResult(),
0.001);
}
```

Nr. 4

```
@Test
public void throwFaultTest() throws Exception {
    StatisticsServiceImpl service = new StatisticsServiceImpl();

    Request request = createRequest(Arrays.asList(Double.NaN));

    try {
        service.calculateStandardDeviation(request);
        fail();
    } catch (Fault e) { }
}
```

Hieronder de uitkomst van de testen, inclusief de integratietest:

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running nl.hu.iac.ws.StatisticsServiceImplTest
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.082 sec
```

6.Integratietest

Er is ook een integratietest geïmplementeerd met behulp van JUnit 4.

Nr. 5

```
@Test
public void integrationTest() throws Exception {
    calculateMean();
    calculateVariance();
    calculateStandardDeviation();

    checkLastResult(expectedStandardDeviation);
}
```

Hiermee wordt gecontroleerd of na het uitvoeren van meerdere berekeningen de checkLastResult functie werkt.

7.Regressietests

Doordat de unit testen automatisch worden uitgevoerd bij deployment zullen deze ook dienen als regressietests.