Eric Boot

SIE-V2A

11-03-2017

TesTPlan Praktijkopdracht 1

Versie 1.0

Inhoudsopgave

[1 Versie 3](#_Toc477028535)

[2 Introductie 3](#_Toc477028536)

[3 Tests 3](#_Toc477028537)

[3.1 Integratie test 3](#_Toc477028538)

[3.2 Unittest 3](#_Toc477028539)

[3.3 Smoketest 5](#_Toc477028540)

# 1 Versie

|  |  |  |
| --- | --- | --- |
| **Versie** | **Opmerking** | **Testplan-versie** |
| 1.0 | Definitieve testplan | 1.0 |
|  |  |  |

# 2 Introductie

Dit testplan is onderdeel van de 1e praktijkopdracht van de cursus Intigration and Communcation. Voor deze opdracht wordt er verwacht dat de student een SOAP Service realiseert in het thema ‘the Smart Cities’.

De test die zullen worden uitgevoerd zijn: Integratie test, Unittest en Smoketest. Deze tests zijn verplicht gesteld in de opdracht.

# 3 Tests

## 3.1 Integratie test

De integratie test is niet aanwezig, omdat er geen pom.xml bestaat in het project.

## 3.2 Unittest

Voor de unittest is er gebruik gemaakt van jUnit. Met dit framework worden er tests gerund die tonen dat de uitkomst correct is. Hieronder is de source code te vinden.

|  |
| --- |
| **package** test;  **import** customer.\*; **import** org.junit.Test; **import** producer.WindServiceImpl;  **import** javax.xml.datatype.DatatypeConfigurationException; **import** javax.xml.datatype.DatatypeFactory; **import** javax.xml.datatype.XMLGregorianCalendar; **import** java.text.DateFormat; **import** java.text.ParseException; **import** java.text.SimpleDateFormat; **import** java.util.Date; **import** java.util.GregorianCalendar; **public class** jUnit {   @Test  **public void** doTest1() **throws** Fault\_Exception, DatatypeConfigurationException, ParseException {  Request request = **new** Request();  DateTime dateTime = **new** DateTime();  Location location = **new** Location();   *//set dateTime* DateFormat dformat = **new** SimpleDateFormat(**"yyyy-mm-dd"**);  Date date = dformat.parse(**"2017-02-02"**);  GregorianCalendar cDate = **new** GregorianCalendar();  cDate.setTime(date);  XMLGregorianCalendar date2 = DatatypeFactory.*newInstance*().newXMLGregorianCalendar(cDate);   DateFormat tformat = **new** SimpleDateFormat(**"hh:mm:ss"**);  Date time = tformat.parse(**"09:00:00"**);  GregorianCalendar cTime = **new** GregorianCalendar();  cTime.setTime(time);  XMLGregorianCalendar time2 = DatatypeFactory.*newInstance*().newXMLGregorianCalendar(cTime);   dateTime.setDate(date2);  dateTime.setTime(time2);   request.setDateTime(dateTime);   *//set location* location.setLatitude(89.111);  location.setLongitude(-11.11);   request.setLocation(location);   WindServiceImpl windService = **new** WindServiceImpl();  Response response = windService.getWindSpeed(request);    System.***out***.println(response.getWindspeed().toString() + response.getUnit().toString());   }   @Test  **public void** doTest2() **throws** Fault\_Exception, DatatypeConfigurationException, ParseException {  Request request = **new** Request();  DateTime dateTime = **new** DateTime();  Location location = **new** Location();   *//set dateTime* DateFormat dformat = **new** SimpleDateFormat(**"yyyy-mm-dd"**);  Date date = dformat.parse(**"2017-12-02"**);  GregorianCalendar cDate = **new** GregorianCalendar();  cDate.setTime(date);  XMLGregorianCalendar date2 = DatatypeFactory.*newInstance*().newXMLGregorianCalendar(cDate);   DateFormat tformat = **new** SimpleDateFormat(**"hh:mm:ss"**);  Date time = tformat.parse(**"19:00:00"**);  GregorianCalendar cTime = **new** GregorianCalendar();  cTime.setTime(time);  XMLGregorianCalendar time2 = DatatypeFactory.*newInstance*().newXMLGregorianCalendar(cTime);   dateTime.setDate(date2);  dateTime.setTime(time2);   request.setDateTime(dateTime);   *//set location* location.setLatitude(1.111);  location.setLongitude(-111.11);   request.setLocation(location);   WindServiceImpl windService = **new** WindServiceImpl();  Response response = windService.getWindSpeed(request);   System.***out***.println(response.getWindspeed().toString() + response.getUnit().toString());    }  } |

De testcases die behoren bij de unittest zijn er 2:

|  |  |  |  |
| --- | --- | --- | --- |
| Testcase | Code naam | Versie 1.0 | Opmerking |
| 1 | doTest1 |  |  |
| 2 | doTest2 |  |  |

## 3.3 Smoketest

Er is een Smoketest aanwezig bij dit project. Deze is aanwezig in de file SmokeTest.java, te vinden in de test package van de source code. De werking van deze test is, dat er kan worden aangetoond dat het werkt en niet error meldingen teruggeeft die ervoor zorgen dat de service niet meer werkt.