Testplan

Auteur
Jonathan Peeman

Datum
18 juni 2017

Versie
0.1

Versie

0.1

Start document

Inhoudsopgave

Introductie

Voor het project ‘Integration & Communication (TCIF-V2IAC1-15)’ moet er een testplan gemaakt worden die uitgevoerd gaat worden. Er wordt een zelfgemaakte SOAP service getest.

Er is een service gemaakt voor het ophalen van een adres en postcode. De service maakt gebruik van de website PostNL.

De volgende tests worden uitgevoerd:

- Smoketest
- Integratietest
- Unit tests
- Regressietests

Testomgeving

De service die getest gaat worden heeft de naam ‘AddressService’. De volgende versie wordt getest: https://github.com/thec0mpler/HU-soap-service.

De volgende middelen worden gebruikt voor het testen van de service:

- SoapUI 5.3.0
- JUnit

Smoketest

De request-berichten die in de map ‘messages’ staan moeten uitgevoerd worden. Als de service altijd met een SOAP bericht antwoord kan er verder getest worden.

Integratietest

De volgende operaties worden aangeboden:

- searchAddress
- searchPostcode

Operatie: searchAddress

De volgende gegevens moeten meegestuurd worden:

- postcode
- huisnummer

De volgende gegevens moeten worden teruggestuurd:

- straat
- huisnummer
- huisnummer toevoeging
- postcode
- plaats

Operatie: searchPostcode

De volgende gegevens moeten meegestuurd worden:

- straat
- huisnummer
- huisnummer toevoeging
- plaats

De volgende gegevens moeten worden teruggestuurd:

- straat
- huisnummer
- huisnummer toevoeging
- postcode
- plaats

Unit tests

Api test

```
import org.junit.Test;
import postcode.AddressType;
import postcode.Api;

import java.io.IOException;

import static org.junit.Assert.assertEquals;

public class ApiTest {
    @Test
    public void getAddressHome() throws IOException {
        AddressType address = Api.getAddress("3824AH", 42, "");

        assertEquals("", address.getAddition());
        assertEquals("AMERSFOORT", address.getPlace());
        assertEquals("3824AH", address.getPostcode());
        assertEquals("Notarisappelgaarde", address.getStreet());
        assertEquals(42, address.getHouseNumber());
    }

    @Test
    public void getAddressDL200() throws IOException {
        AddressType address = Api.getAddress("3584BJ", 200, "");

        assertEquals("", address.getAddition());
```

```
        assertEquals("UTRECHT", address.getPlace());
        assertEquals("3584BJ", address.getPostcode());
        assertEquals("Daltonlaan", address.getStreet());
        assertEquals(200, address.getHouseNumber());
    }

    @Test
    public void getPostcodeHome() throws IOException {
        AddressType address =
Api.getPostcode("Notarisappelgaarde", 42, "", "Amersfoort");

        assertEquals("", address.getAddition());
        assertEquals("Amersfoort", address.getPlace());
        assertEquals("3824AH", address.getPostcode());
        assertEquals("Notarisappelgaarde", address.getStreet());
        assertEquals(42, address.getHouseNumber());
    }

    @Test
    public void getPostcodeDL200() throws IOException {
        AddressType address = Api.getPostcode("Daltonlaan", 200,
"", "Utrecht");

        assertEquals("", address.getAddition());
        assertEquals("Utrecht", address.getPlace());
        assertEquals("3584BJ", address.getPostcode());
        assertEquals("Daltonlaan", address.getStreet());
        assertEquals(200, address.getHouseNumber());
```

```
    }
}
```

Postcode implementatie test

```
import org.junit.Test;

import postcode.InputFault_Exception;

import postcode.PostcodeImpl;


import javax.xml.ws.Holder;


import static org.junit.Assert.assertTrue;


public class PostcodeImplTest {

    @Test

    public void searchAddressHome() {

        try {

            PostcodeImpl postcode = new PostcodeImpl();

            postcode.searchAddress(new Holder("3824AHH"), new
Holder(42), new Holder("Notarisappelgaarde"), new Holder(""), new
Holder("Amersfoort"));

        } catch (InputFault_Exception e) {

            assertTrue(true);


            return;

        }


        assertTrue(false);

    }
```

```
    @Test
    public void searchAddressDL200() {
        try {
            PostcodeImpl postcode = new PostcodeImpl();
            postcode.searchAddress(new Holder("3584BJJ"), new
Holder(200), new Holder("Daltonlaan"), new Holder(""), new
Holder("Utrecht"));
        } catch (InputFault_Exception e) {
            assertTrue(true);

            return;
        }

        assertTrue(false);
    }

    @Test
    public void searchPostcodeHome() {
        try {
            PostcodeImpl postcode = new PostcodeImpl();
            postcode.searchPostcode(new
Holder("Notarisappelgaarde"), new Holder(420), new Holder(""), new
Holder("Amersfoort"), new Holder("3824AH"));
        } catch (InputFault_Exception e) {
            assertTrue(true);

            return;
        }
```

```
        assertTrue(false);
    }

    @Test
    public void searchPostcodeDL200() {
        try {
            PostcodeImpl postcode = new PostcodeImpl();
            postcode.searchPostcode(new Holder("Daltoonlaan"), new
Holder(2001), new Holder(""), new Holder("Utrecht"), new
Holder("3584BJ"));
        } catch (InputFault_Exception e) {
            assertTrue(true);

            return;
        }

        assertTrue(false);
    }
}
```

Regressietests

Voor de regressietests kunnen de berichten uit de map ‘messages’ gebruikt worden.