

Testplan

Auteur

Jonathan Peeman

Datum

20 juni 2017

Versie

0.1

Versie

0.1

Start document

Inhoudsopgave

Versie.....	2
0.1.....	2
Introductie.....	4
Testomgeving.....	5
Smoketest.....	6
Integratietest.....	7
Operatie: searchAddress.....	7
Operatie: searchPostcode.....	7
Unit tests.....	8
Api test.....	8
Postcode implementatie test.....	10
Regressietests.....	13

Introductie

Voor het project 'Integration & Communication (TCIF-V2IAC1-15)' moet er een testplan gemaakt worden die uitgevoerd gaat worden. Er wordt een zelfgemaakte REST 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 'HU-rest-service'. De volgende versie wordt getest: <https://github.com/thec0mpler/HU-rest-service>.

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

- Firefox addon: HttpRequester 2.2
- JUnit

Smoketest

De request-berichten die in de map 'messages' staan moeten uitgevoerd worden. Als de service altijd met een JSON 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.Address;
import postcode.Api;

import java.io.IOException;

import static org.junit.Assert.assertEquals;

public class ApiTest {
    @Test
    public void getAddressHome() throws IOException {
        Address 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 {
        Address 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 {
    Address 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 {
    Address 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.PostcodeController;  
  
import java.io.IOException;  
  
import static org.junit.Assert.assertTrue;  
  
public class PostcodeImplTest {  
    @Test  
    public void searchAddressHome() {  
        try {  
            PostcodeController postcode = new  
PostcodeController();  
            postcode.searchAddress("3824AHH", 42, "");  
        } catch (IOException e) {  
            assertTrue(true);  
  
            return;  
        }  
  
        assertTrue(false);  
    }  
  
    @Test  
    public void searchAddressDL200() {
```

```

        try {
            PostcodeController postcode = new
PostcodeController();
            postcode.searchAddress("3584BJJ", 200, "");
        } catch (IOException e) {
            assertTrue(true);

            return;
        }

        assertTrue(false);
    }

    @Test
    public void searchPostcodeHome() {
        try {
            PostcodeController postcode = new
PostcodeController();
            postcode.searchPostcode("Notarisappelgaarde", 420, "",
"Amersfoort");
        } catch (IOException e) {
            assertTrue(true);

            return;
        }

        assertTrue(false);
    }

```

```

@Test
public void searchPostcodeDL200() {
    try {
        PostcodeController postcode = new
PostcodeController();
        postcode.searchPostcode("Daltoonlaan", 2001, "",
"Utrecht");
    } catch (IOException e) {
        assertTrue(true);

        return;
    }

    assertTrue(false);
}
}

```

Regressietests

Voor de regressietests kunnen de berichten uit de map 'messages' gebruikt worden.