# Versie

1.0

# Introductie

De parkeerplaatsservice is bedoeld voor een parkeerplaats. Hij houdt bij hoeveel parkeerplekken er nog vrij zijn en hoeveel al bezet en welke auto’s er zijn geparkeerd.

Ik zal een smoketest, black box test en unit tests doen.

# Testomgeving

Ik ga de services ‘getAantalparkeerplaatsen’, ‘nieuweAuto’, ‘getGeparkeerdeAutos’ en ‘verwijderAuto’ testen. Voor de unit testen zal ik Junit gebruiken en voor black box tests Soapui

# Smoketest

‘Does the program run?’ Ja hij draait, dit is te zien op: <https://parkeerplaatsservice-lbos.rhcloud.com/services/getParkeerplaatsen>

# Testcases black box test

De operations:

<operation name="getAantalParkeerplaatsen">...</operation>

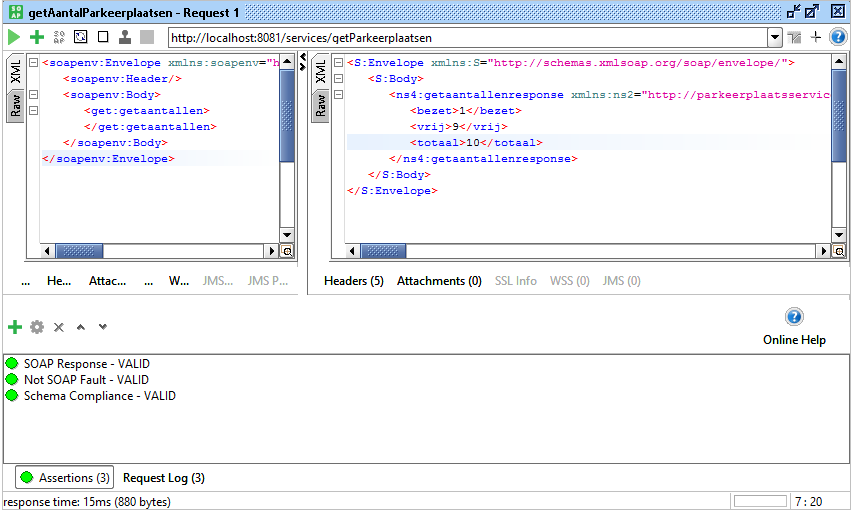
<operation name="nieuweAuto">...</operation>

<operation name="getGeparkeerdeAutos">...</operation>

<operation name="verwijderAuto">...</operation>

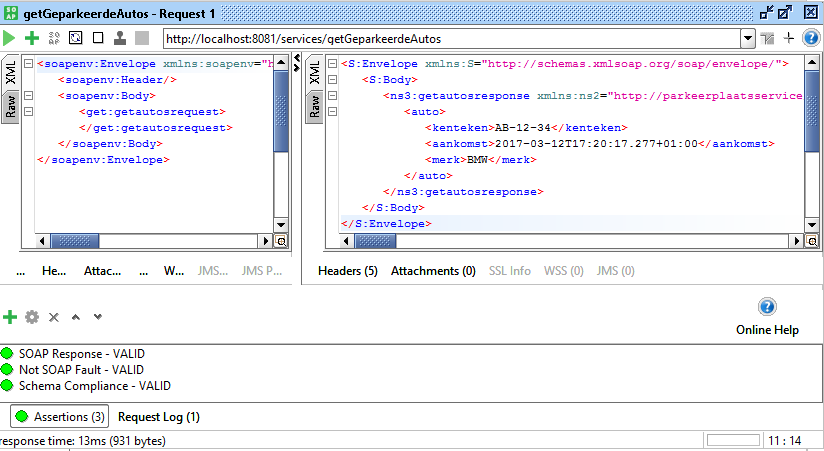
## getAantalParkeerplaatsen

Vragen naar het aantal parkeerplaatsen. Verwacht als resultaat te zien hoeveel parkeerplekken bezet en hoeveel vrij zijn.



## getGeparkeerdeAuto’s

Vragen naar de geparkeerde auto’s. Ik verwacht als resultaat een lijst met de geparkeerde auto’s. Per auto de tijd en datum dat hij binnen kwam, het kenteken en het merk.



## nieuweAuto

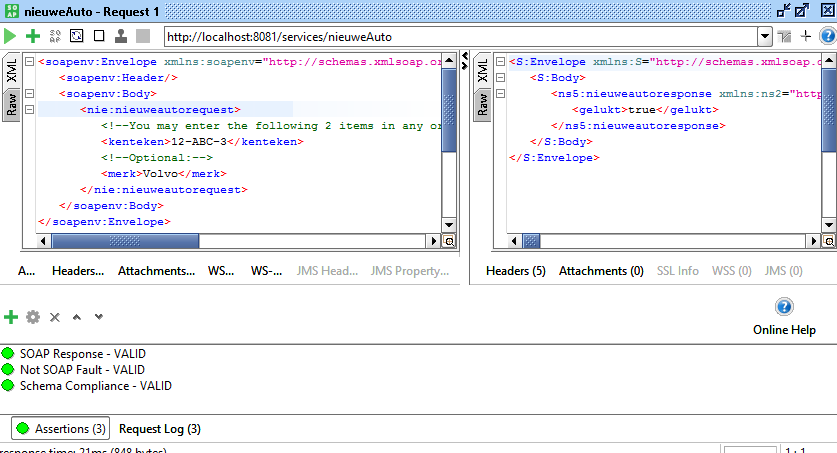
Een nieuwe auto toevoegen aan de parkeerplaats. Het aantal vrije plaatsen zal één naar beneden gaan het aantal bezette plaatsen één omhoog. Wanneer je de geparkeerde auto’s opvraagt moet de nieuwe auto er ook tussen staan.

Aantal vrije parkeerplaatsen voor nieuwe auto: 9

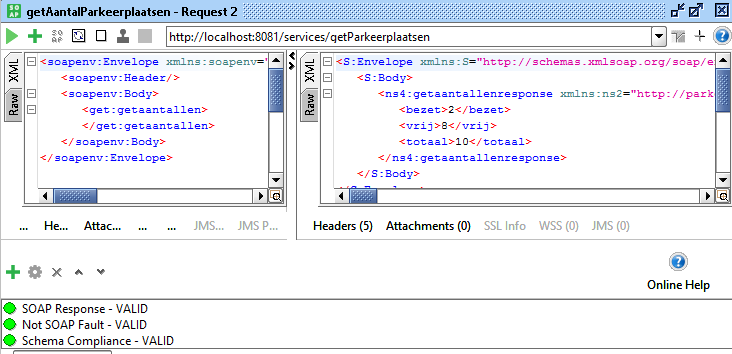
Aantal bezette parkeerplaatsen voor nieuwe auto: 1

Get geparkeerde auto’s geeft 1 auto (zie vorige test)

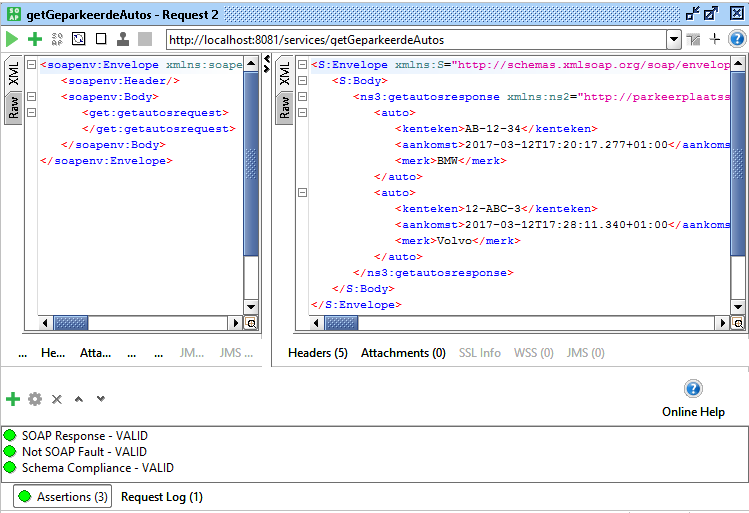
Voeg nieuwe auto toe:



getAantalParkeerplaatsen na toevoegen:



Get geparkeerde auto’s na toevoegen:

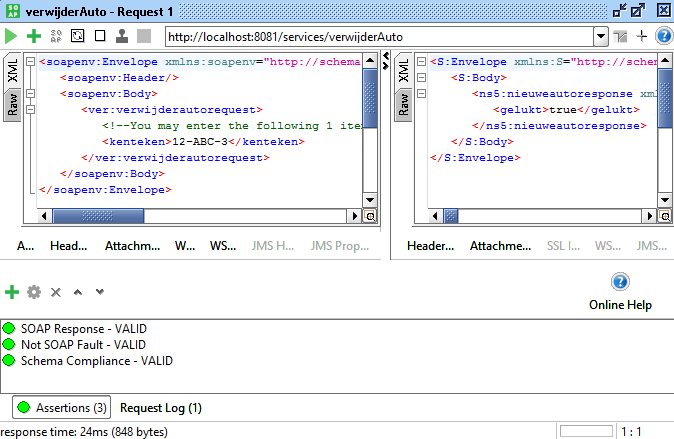


## Verwijderauto

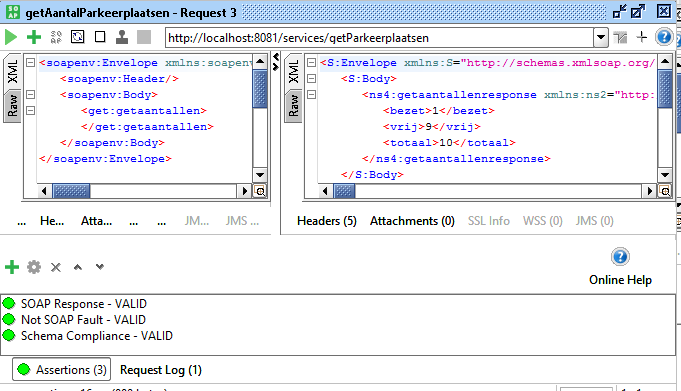
Een auto verwijderen a.d.h.v. zijn kenteken. Het aantal vrije plaatsen moet één omhooggaan en aantal bezette parkeerplaatsen één naar beneden. Je zal de auto ook niet meer moeten zien als je de geparkeerde auto’s opvraagt.

Zie vorige test voor aantal vrije parkeerplaatsen, bezette parkeerplaatsen en geparkeerd auto’s.

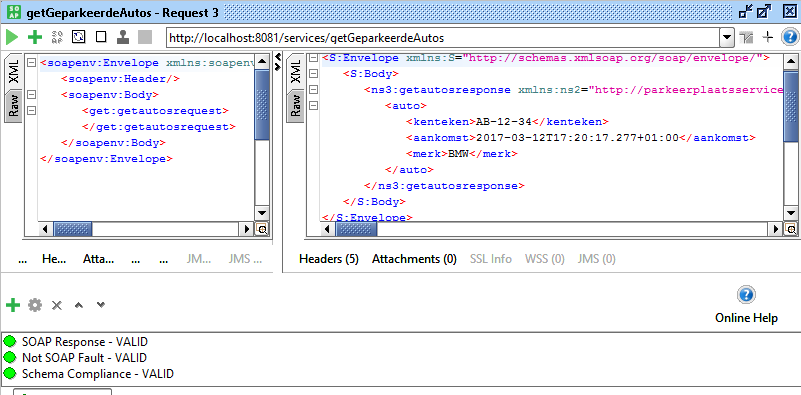
Auto verwijderen:

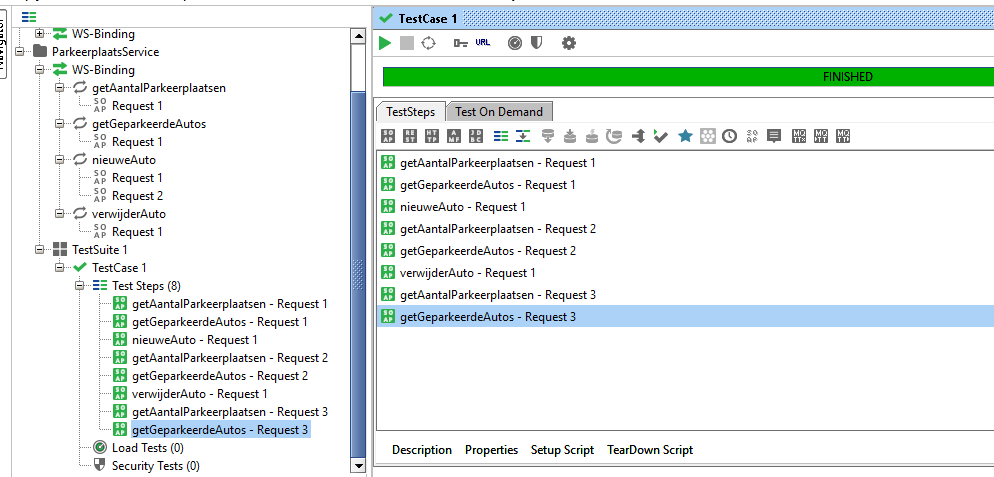


Get aantal parkeerplaatsen na verwijderen:



Get geparkeerde auto’s na verwijderen:





# Unit test

Voor de unit tests gebruik ik JUnit.

Elke operatie die de service aanbiedt wordt hiermee getest.

Source code:

**import** model.JsonConverter;  
**import** model.Parkeerplaats;  
**import** model.ParkeerplaatsServiceImpl;  
**import** org.junit.Before;  
**import** org.junit.Test;  
**import** parkservice.\*;  
  
**import** java.util.concurrent.ThreadLocalRandom;  
  
**import static** org.junit.Assert.assertTrue;  
  
**public class** TestParkeerplaatsService {  
 **private int aantalVrij**;  
 **private int aantalBezet**;  
 **private** Parkeerplaats **parkeerplaats**;  
 **private** ObjectFactory **factory** = **new** ObjectFactory();  
 **private** ParkeerplaatsServiceImpl **parkeerplaatsService** = **new** ParkeerplaatsServiceImpl();  
  
 @Before  
 **public void** setUp(){  
 **parkeerplaats** = JsonConverter.*getParkeerplaats*();  
 **aantalVrij** = **parkeerplaats**.getVrij();  
 **aantalBezet** = **parkeerplaats**.getBezet();  
 }  
  
 @Test  
 **public void** getAantalParkeerplaatsen(){*//kijk of totaal = bezet + vrij* Getaantallenrequest request = **factory**.createGetaantallenrequest();  
 Getaantallenresponse response = **parkeerplaatsService**.getAantalParkeerplaatsen(request);  
 *assertTrue*(response.getTotaal().equals(response.getBezet().add(response.getVrij())));  
 }  
  
 @Test  
 **public void** testNieuweAuto() **throws** Exception {  
 Nieuweautorequest request = **factory**.createNieuweautorequest();  
 request.setKenteken(**"xx-00-xx"**);  
 request.setMerk(Automerk.*values*()[ThreadLocalRandom.*current*().nextInt(0, Automerk.*values*().**length**)]);  
 **parkeerplaatsService**.nieuweAuto(request);  
 **parkeerplaats** = JsonConverter.*getParkeerplaats*();  
 *assertTrue*(**parkeerplaats**.getVrij() == **aantalVrij** - 1 && **parkeerplaats**.getBezet() == **aantalBezet** + 1);  
 }  
  
 @Test  
 **public void** testVerwijderAuto() **throws** Exception {  
 Verwijderautorequest request = **factory**.createVerwijderautorequest();  
 request.setKenteken(**"xx-00-xx"**);  
 **parkeerplaatsService**.verwijderAuto(request);  
 **parkeerplaats** = JsonConverter.*getParkeerplaats*();  
 *assertTrue*(**parkeerplaats**.getVrij() == **aantalVrij** + 1 && **parkeerplaats**.getBezet() == **aantalBezet** - 1);  
 }  
  
 @Test  
 **public void** testGetGeparkeerdeAutos() {  
 Getautosrequest request = **factory**.createGetautosrequest();  
 Getautosresponse response = **parkeerplaatsService**.getGeparkeerdeAutos(request);  
 *assertTrue*(response.getAuto().size() == **parkeerplaats**.getBezet());  
 }  
  
}

