Inhoud

[Black box tests 1](#_Toc478234596)

[Test case 1. 1](#_Toc478234597)

[Test case 2. 2](#_Toc478234598)

[Test case 3. 2](#_Toc478234599)

[Test case 4. 3](#_Toc478234600)

[Test case 5. 3](#_Toc478234601)

[Test case 6. 4](#_Toc478234602)

[Test case 7. 4](#_Toc478234603)

[Test case 8. 5](#_Toc478234604)

[Test case 9. 5](#_Toc478234605)

[Uitslag: 6](#_Toc478234606)

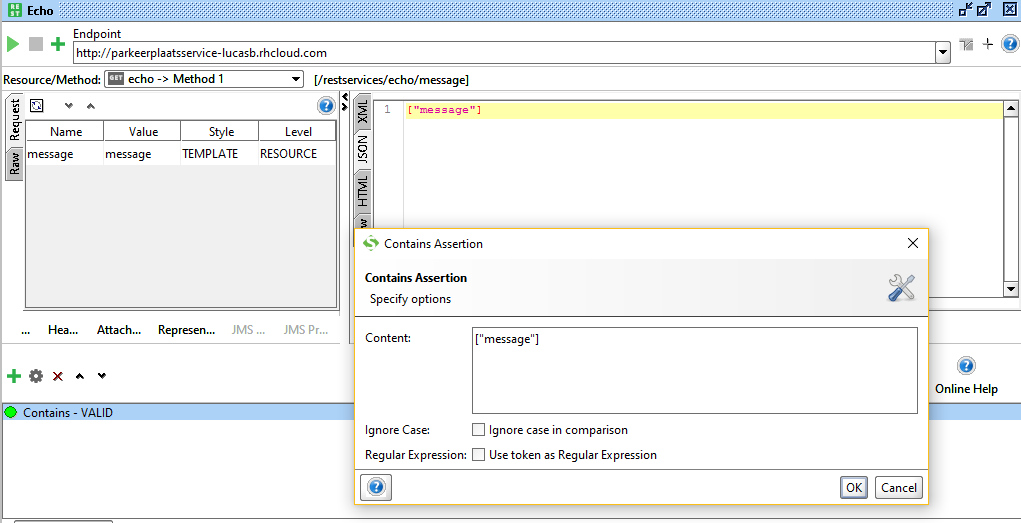
[Unit tests 6](#_Toc478234607)

[Code 6](#_Toc478234608)

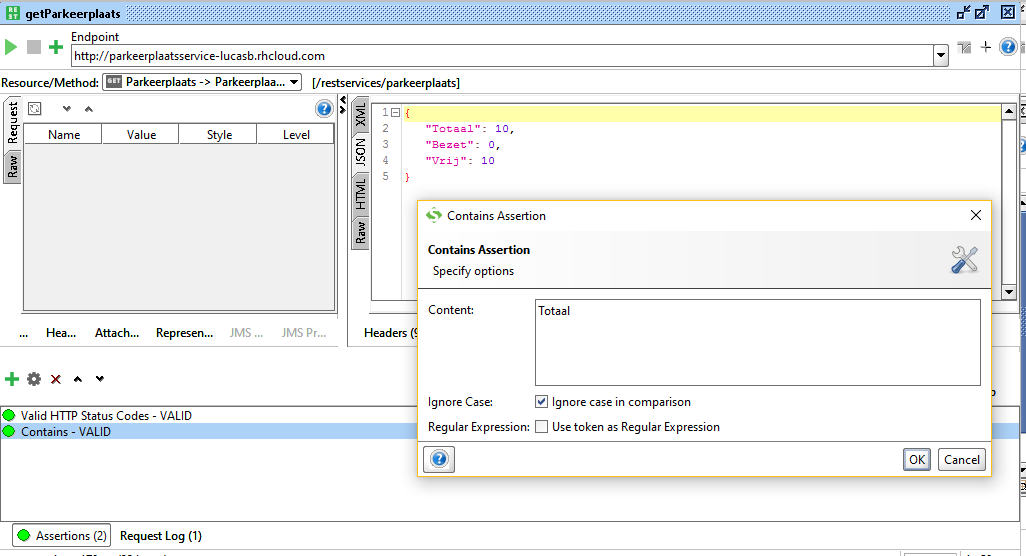
[Uitslag: 8](#_Toc478234609)

# Black box tests

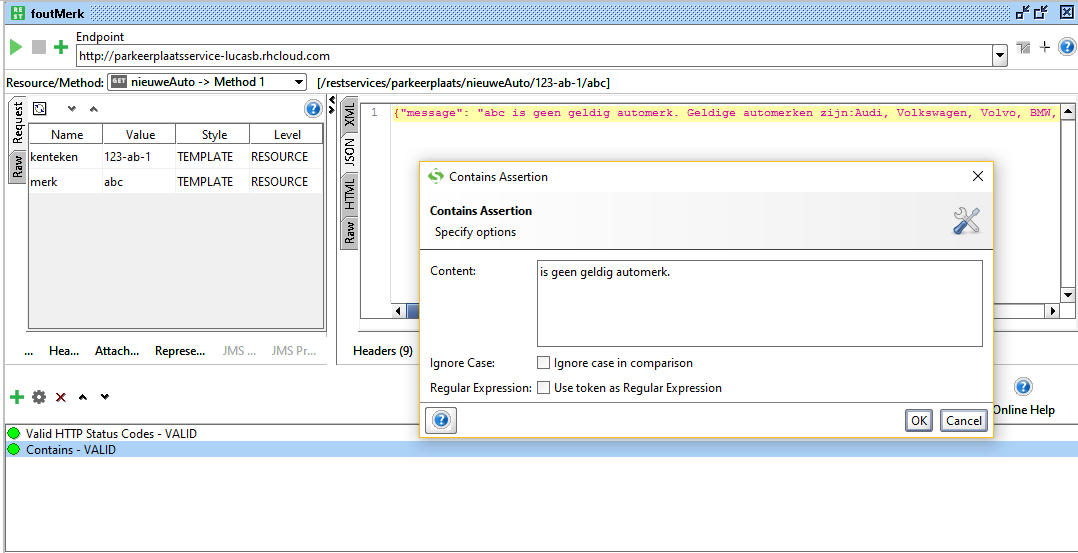
## Test case 1.



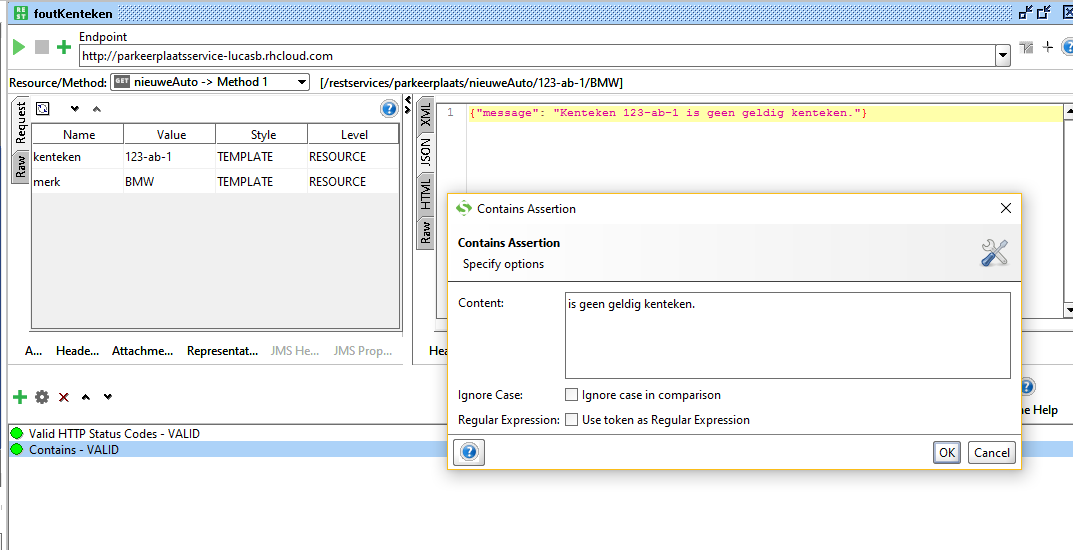
## Test case 2.



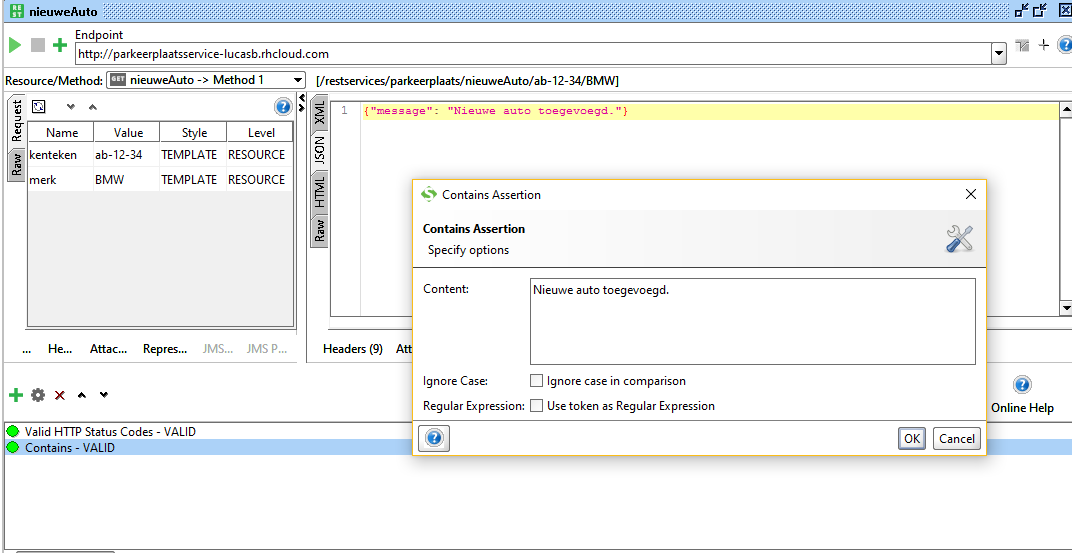
## Test case 3.



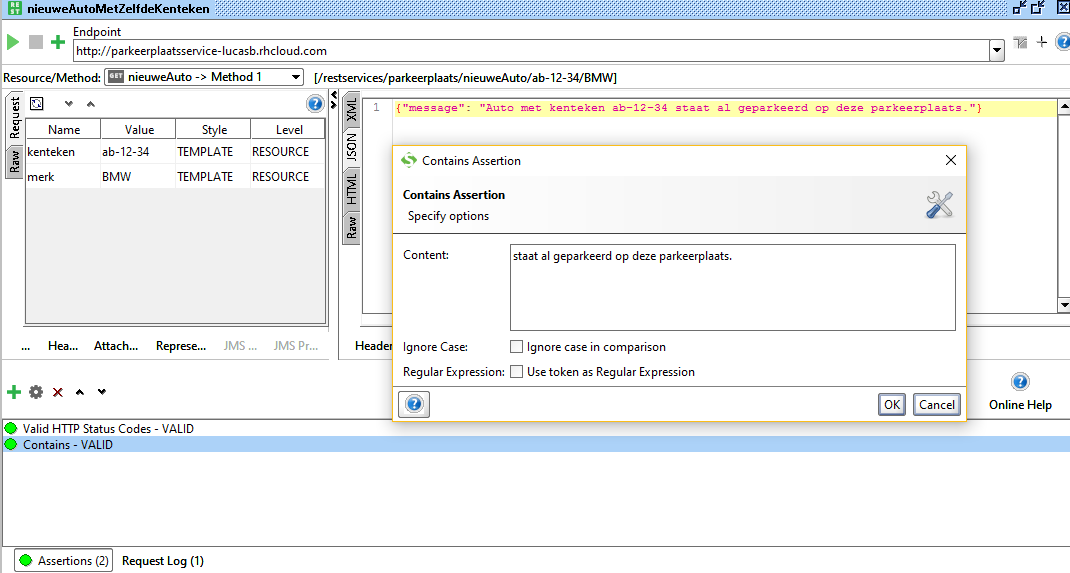
## Test case 4.



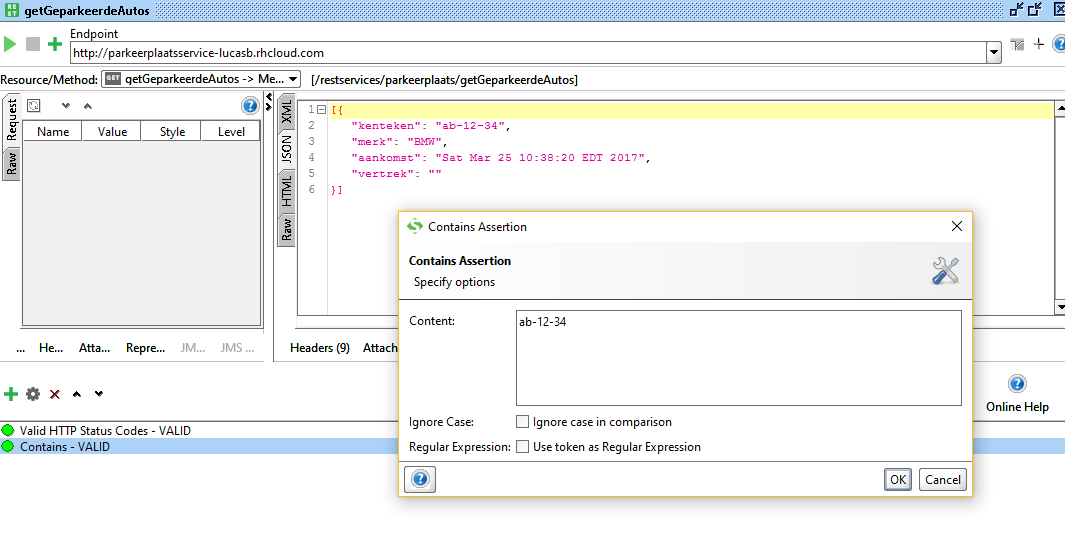
## Test case 5.



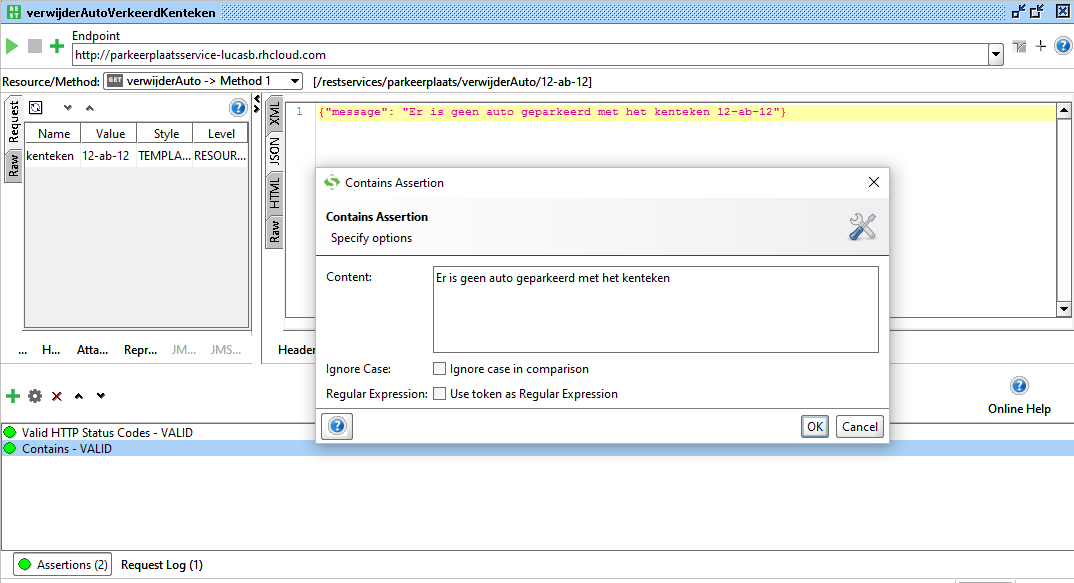
## Test case 6.



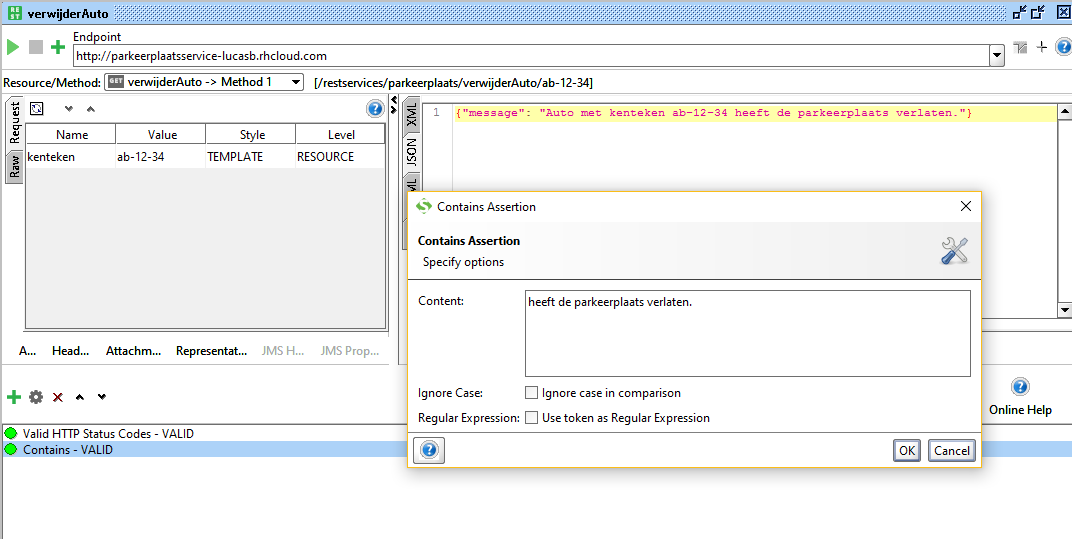
## Test case 7.



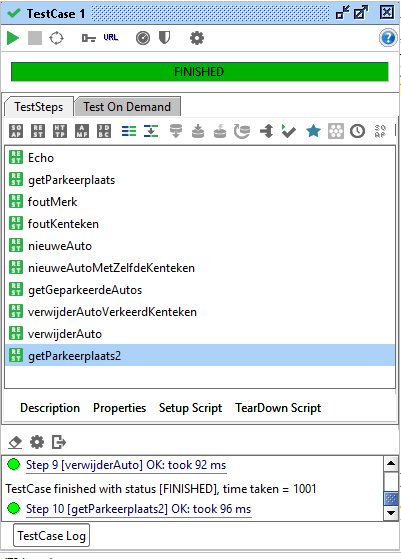
## Test case 8.



## Test case 9.



## Uitslag:



# Unit tests

## Code

**import** com.google.gson.Gson;  
**import** com.google.gson.GsonBuilder;  
**import** model.Automerk;  
**import** model.Parkeerplaats;  
**import** org.junit.Test;  
**import** org.junit.FixMethodOrder;  
**import** org.junit.runners.MethodSorters;  
**import** resources.EchoService;  
**import** resources.ParkeerplaatsService;  
  
**import static** org.junit.Assert.assertTrue;  
  
@FixMethodOrder(MethodSorters.***NAME\_ASCENDING***)  
**public class** TestParkeerplaatsService {  
 EchoService **echoService** = **new** EchoService();  
 ParkeerplaatsService **parkeerplaatsService** = **new** ParkeerplaatsService();  
 GsonBuilder **builder** = **new** GsonBuilder();  
 Gson **gson** = **builder**.create();  
 String **kenteken** = **"ab-12-34"**;  
 String **merk** = **"BMW"**;  
  
 @Test  
 **public void** t1\_Echo(){  
 String message = **"hello"**;  
 *assertTrue*(**echoService**.echoMessage(message).contains(message));  
 }  
  
 @Test  
 **public void** t2\_getParkeerplaats(){*//kijk of totaal = bezet + vrij* Parkeerplaats p = **gson**.fromJson(**parkeerplaatsService**.getParkeerplaats(), Parkeerplaats.**class**);  
 *assertTrue*(p.getTotaal() == (p.getBezet() + p.getVrij()));  
 }  
  
 @Test()  
 **public void** t3\_testFoutMerk(){  
 String merk = **"abc"**;  
 *assertTrue*(**parkeerplaatsService**.nieuweAuto(**kenteken**, merk).contains(merk + **" is geen geldig automerk"**));  
 }  
  
 @Test  
 **public void** t4\_testFoutKenteken(){  
 String kenteken = **"123-ab-1"**;  
 *assertTrue*(**parkeerplaatsService**.nieuweAuto(kenteken, Automerk.*values*()[0].name()).contains(**"Kenteken "** + kenteken + **" is geen geldig kenteken"**));  
 }  
  
 @Test  
 **public void** t5\_testNieuweAuto(){  
 *assertTrue*(**parkeerplaatsService**.nieuweAuto(**kenteken**, **merk**).contains(**"Nieuwe auto toegevoegd."**));  
 }  
  
 @Test  
 **public void** t6\_testNieuweAutoZelfdeKenteken(){  
 *assertTrue*(**parkeerplaatsService**.nieuweAuto(**kenteken**, **merk**).contains(**"Auto met kenteken "** + **kenteken** + **" staat al geparkeerd op deze parkeerplaats."**));  
 }  
  
 @Test  
 **public void** t7\_testGetGeparkeerdeAutos(){  
 String response = **parkeerplaatsService**.getGeparkeerdeAutos();  
 *assertTrue*(response.contains(**kenteken**) && response.contains(**merk**));  
 }  
  
 @Test  
 **public void** t8\_testVerwijderAutoVerkeerdKenteken(){  
 String kentekentemp = **"12-ab-12"**;  
 *assertTrue*(**parkeerplaatsService**.verwijderAuto(kentekentemp).contains(**"Er is geen auto geparkeerd met het kenteken "** + kentekentemp));  
 }  
  
 @Test  
 **public void** t9\_verwijderAuto(){  
 *assertTrue*(**parkeerplaatsService**.verwijderAuto(**kenteken**).contains(**"Auto met kenteken "** + **kenteken** + **" heeft de parkeerplaats verlaten."**));  
 }  
}

## Uitslag:

