**Testplan**

|  |  |
| --- | --- |
| **Naam:** | Serkan Sevinc |
| **Klas:** | V2C |
| **Datum:** | 09-06-2017 |
| **Service:** | Koffieautomaten service |

Inhoudsopgave

[Versie 3](#_Toc484807258)

[Introductie 3](#_Toc484807259)

[Testomgeving 3](#_Toc484807260)

[Smoketest 4](#_Toc484807261)

[Testcases black box test (integratietest of functionele test) 5](#_Toc484807262)

[Unit Tests 6](#_Toc484807263)

[Regressietests 7](#_Toc484807264)

## Versie

|  |  |  |  |
| --- | --- | --- | --- |
| **Nummer** | **Gewijzigd door** | **Datum** | **Commentaar** |
| 1.0 | Serkan Sevinc | 09-06-2017 | Eerste versie |

## Introductie

Mijn service gaat over de koffieautomaten in Smart Cities, het bied services aan om de alle koffieautomaten te zien, te zoeken op code en overzicht krijgen van de top 3 koffieautomaten. De volgende tools worden gebruikt voor de applicatie: JUNIT, Gson, jQuery, JSON, Maven en Jersey. JUNIT wordt dan gebuikt voor de unittest, Maven als integratie test, Jersey als de web framework, met Gson om java objecten te serialiseren van (en naar) JSON en jQuery voor het inlezen van data op HTML.

## Testomgeving

Volgende tools: jUnit, Spark, Gson Maven. jUnit wordt dan gebuikt voor de unittest, Maven als integratie test, Jersey als framework voor REST en Gson als Java library.

|  |  |
| --- | --- |
| **jUnit** | Unittest |
| **Maven** | Integratie test, download de benodigde JAR bestanden van de applicatie. In mijn voor geval jUnit. |
| **Gson** | Open-source Java library |
| **Jersey** | Framework voor REST |
| **jQuery** | Inlezen van JSON data op HTML |

## Smoketest

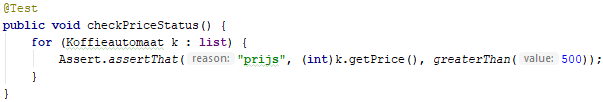
Voor deze test heb ik met jUnit gebruikt. Om bepaalde functies te testen zoals hoe de actuele parameter groter is dan de aangegeven parameter of om de parameters met elkaar te vergelijken heb ik van **Hamcrest** de **Matchers** gebruikt:



Allereerst wordt de service opgeroepen om de methodes bereikbaar te maken zoals hieronder:



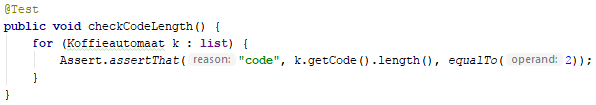
De eerste test is om elke prijs van het lijst af te gaan en het te vergelijken met een nummer, zoals je hier onder ziet heb ik daarvoor ook **assertThat** functie van jUnit gebruikt en de **greaterThan** functie van **Hamcrest**. Bij het uitvoeren van de Test krijg je dan ook een error omdat niet elke prijs hoger is dan 500:



Resultaat:



Bij de volgende test wordt er gekeken of de code van elke koffieautomaat gelijk is aan 2, want dat is namelijk de business rule ook hierbij **assertThat** functie maar dan **equalTo** van **Hamcrest**:



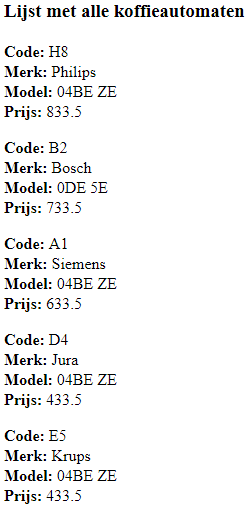
Resultaat bij code lengte dat verschilt dan 2:



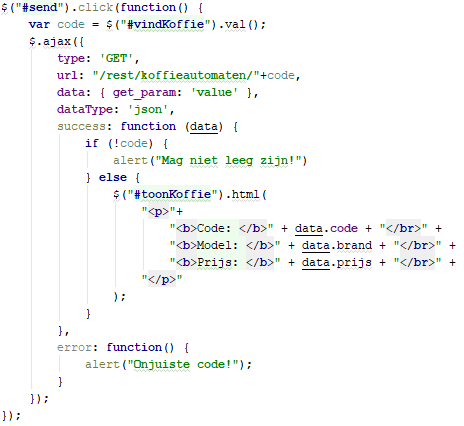
## Testcases black box test (integratietest of functionele test)

Voor REST heb ik Jersey gebruikt om bij paden JSON data te tonen en jQuery gebruikt om dat in te lezen in HTML. Zoals je hieronder ziet, heb je de eerst functie van zoeken. Bij het zoeken van koffieautomaten met code krijg je de betreffende informatie te zien. Volgende functie is het tonen van top 3 automaten en daarna het lijst met alle koffieautomaten. Dit krijg je gewoon bij de root path.

Als je puur JSON formaat wil zien zit dat dan bij: **/rest/koffieautomaten**, **/rest/koffieautomaten/<id>** en **/rest/koffieautomaten/prijzigste**. Voor de top 3 functie is er een **Comparator** gebruikt voor de prijzen met **sublist**.

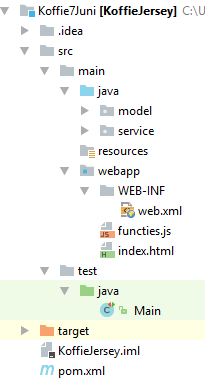
 

jQuery code voor de zoek-functie. Elke functie heeft dan een AJAX methode van jQuery om de JSON in te lezen en te manipuleren in HTML (elke functie zit in de **functies.js**):

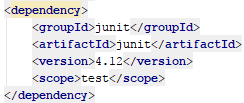


## Unit Tests

Maven maakt automatisch een file structure aan zoals hieronder aangegeven, ik heb ook een **arch-type** gekozen voor de web-app om Jersey werkend te krijgen. Daarmee wordt ook een test map aangemaakt die dan bedoeld is voor het testen. In dat map heb ik mijn jUnit test uitgevoerd.

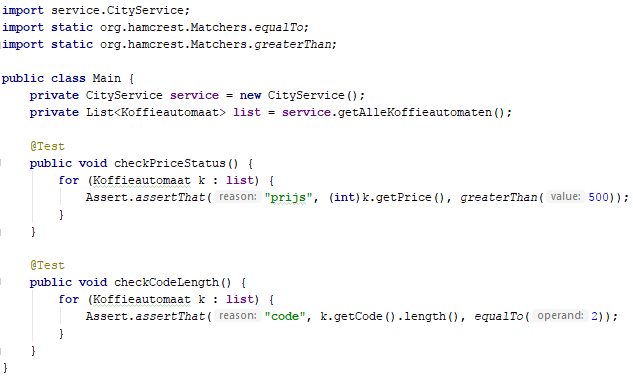


Voor het gebruik maken van jUnit moest ik eerst een dependency aangeven. (Zie hieronder)



Zoals je hieronder ziet wordt de service opgeroepen en de **Hamcrest** functies gebruikt.

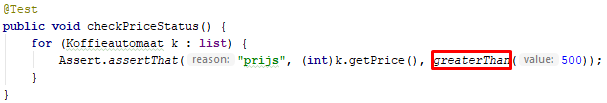
Source code:



## Regressietests

1.

Voor het testen of de prijzen van de koffieautomaten wel hoger zijn dan een bepaald bedrag wou ik eerst **assertEquals** gebruiken van jUnit, maar nadat ik zag dat je niet precies kon zien waar het fout ging bij een error, zocht ik voor een betere optie genaamd Hamcrest. Die gebruikt dan **Matchers** waar je functies als **greaterThan** hebt.



2.

Voor het krijgen van input door de gebruiker heb ik een klik-event gebruikt van jQuery. Eerst probeerde ik AJAX de klikfunctie te implementeren maar ik kon de input waarde van de gebruiker niet krijgen. Dus na een tijdje te spelen met code kwam ik op het idee om het hele functie in een klik event te zetten. En dat werkte prima:

