Programmeren 4 – Practicum 2

# Lees dit eerst

In dit practicum ga je werken met Node.js, Git, Heroku en een aantal ondersteunende software tools. Het practicum sluit aan op de inhoud van Training 2.

Voer de opdrachten uit in **groepjes van 2 studenten**.

Aan het eind van elke opdracht test je of je code correct werkt. Maak tijdens het practicum samen met je medestudent van die testjes van alle opdrachten een kort verslag in PDF formaat (1 of 2 A4-tjes). Lever het verslag individueel in via de inleverlink op Blackboard (let op de inleverdeadline op BB).

De docent die het practicum begeleidt komt tijdens het volgende practicum bij elk groepje langs om de uitwerking te bekijken en eventuele vragen te beantwoorden.

We gaan in dit practicum een nieuwe REST webservice maken met Express en die voorzien van testcode met Mocha en Chai. Onze webservice toont recepten afkomstig uit WikiBooks Cookbook (<https://en.wikibooks.org/wiki/Cookbook:Table_of_Contents>).

Na elke opdracht voeg je de nieuwe of gewijzigde files toe aan je lokale git repository:

git add .

git commit –m “<een betekenisvol bericht dat de wijziging beschrijft>”

# Opdracht 1

Maak binnen je werkdirectory voor het Programmeren 4 practicum (die je vorige week hebt gemaakt als het goed is) een nieuwe map aan voor een nieuw WebStorm project. Noem de map bijvoorbeeld ‘MyFirstMochaProject’.

Maak in deze map een nieuw WebStorm project.

Maak voor dit project een package.json aan:

npm init

Installeer Mocha, Chai en Chai-http:

npm install –-save-dev mocha chai chai-http

Installeer Mocha ook globaal (anders kun je mocha niet vanaf de command line starten):

npm install –g mocha

Maak in je projectdirectory een server.js file met een heel eenvoudige REST server die luistert op poort 8080. Kijk naar de practicumopgaven van practicum 1 en naar de presentatie van training 2 hoe dat ook alweer moest.

Test je REST server met een paar requests in Postman.

Voeg de gemaakte files toe aan je git repository. Als het goed is heb je voor je werkdirectory al een git repository gemaakt, met daarin een .gitignore file. Zo niet, doe dat dan alsnog (zie de opdrachten van het vorige practicum voor uitleg).

# Opdracht 2

Maak in je projectdirectory een ‘config.json’ file waarin alle configuratieinstellingen voor je server worden opgeslagen. Zet in deze file:

{

“webPort”: “8080”

}

Kijk in de presentatie van training 2 hoe je deze file kunt inlezen en de “webPort” instelling kunt gebruiken in je server. Pas je server.js file aan om de instellingen uit config.json in te lezen en zorg dat je server het poortnummer uit deze config.json file gebruikt.

Test je server opnieuw, verifieer dat de wijziging werkt. Zet bijvoorbeeld tijdelijk een ander poortnummer in je config.json en test daarmee.

Commit je aanpassingen weer in je lokale git repository. We zullen dat telkens na elke opdracht doen om de ontwikkelgeschiedenis van de server vast te leggen.

# Opdracht 3

We gaan API routes toevoegen aan onze REST server en willen daarbij meteen versies van de API introduceren. We zullen net zoals in de presentatie net doen alsof versie 1 van de API inmiddels achterhaald is.

Maak een map ‘routes’ aan in je projectdirectory en zet daarin twee files ‘routes\_api\_v1.js’ en ‘routes\_api\_v2.js’. In ‘routes\_api\_v1.js’ komt te staan:

// API version 1

var express = require('express');

var router = express.Router();

var path = require('path');

router.get('\*', function(request, response) {

response.status(200);

response.json({

"description": "Recipes REST server API version 1 is no longer supported. Please use API version 2."

});

});

module.exports = router;

Vergeet de laatste regel met module.exports niet, anders werkt je code niet.

In onze server.js file nemen we op:

server.all('\*', function(request, response, next) {

console.log(request.method + " " + request.url);

next();

})

server.use('/api/v1', require('./routes/routes\_api\_v1'));

Start je server en ga na (b.v. met Postman) dat de route naar /api/v1 de verwachte melding geeft.

Commit je wijzigingen.

# Opdracht 4

Vul file ‘routes\_api\_v2.js’ op soortgelijke wijze en neem daarin een route op voor het pad ‘/info’. Laat dat pad een JSON object opleveren met een tekst die je server kort beschrijft. De status code moet 200 worden.

Voeg voor API versie 2 ook een route toe voor ‘\*’ die een status code 404 met een JSON object met daarin een not-found melding oplevert. Kijk in de opdrachten van practicum 1 hoe dat ook alweer ging.

Test je server met Postman, ga na dat beide routes werken.

Commit je wijzigingen.

# Opdracht 5

Voeg in je projectdirectory een file ‘recipes.js’ toe met de volgende inhoud (recepten afkomstig uit <https://en.wikibooks.org/wiki/Cookbook>).

// Sample recipes database

// Source: https://en.wikibooks.org/wiki/Cookbook

var recipes = [

{

name: 'Lasagne',

category: 'Pasta',

procedure: 'Steps for making lasagne'

},

{

name: 'Calzone',

category: 'Pizza',

procedure: 'Steps to make calzone (pizza)'

},

{

name: 'Tuna Salad',

category: 'Salad',

procedure: [

'Mix ingredients in a bowl',

'Chill before serving'

]

}

];

module.exports = recipes;

Zoek zelf op <https://en.wikibooks.org/wiki/Cookbook> je eigen favoriete recept op en voeg dat aan de recipes array in recipes.js toe. Meerdere recepten mag ook.

We moeten in routes\_api\_v2.js nu een route toevoegen die deze recepten oplevert:

/api/v2/recipes 🡪 levert de hele array met recepten

Pas de code zo aan dat dit werkt zoals gevraagd.

Test je code, verifieer dat deze route werkt.

Commit je wijzigingen (en de nieuwe file).

# Opdracht 6

Zoek in de presentatie van deze week en in de repository met code uit de workshop die op BB vermeld staat op hoe je een URL parameter kunt gebruiken in een route. Pas routes\_api\_v2.js zo aan dat de volgende requests werken:

/api/v2/recipes/3 🡪 levert recept nummer 3 uit de array met recepten

/api/v2/recipes/<n> 🡪 levert recept nummer <n>

Je zult hiervoor waarschijnlijk een nieuwe route moeten toevoegen (‘/recipes/:number’).

Test je uitbreidingen met Postman.

Zoek ook op in de presentatie en de repository met voorbeeldcode hoe je een query string kunt gebruiken in een route. Pas wederom routes\_api\_v2.js aan zodanig dat de volgende requests werken:

/api/v2/recipes?category=Salad 🡪 levert alle recepten van category ‘Salad’ uit de array

/api/v2/recipes?category=<cat> 🡪 levert alle recepten van category <cat>

Zoek op internet (b.v. bij w3schools.com of tutorialspoint.com of developer.mozilla.org) hoe de Javascript filter() functie elementen uit een array uitfiltert. Leg uit in een paar zinnen hoe het werkt en waar die callback functie bij filter() voor dient.

Test je uitbreidingen met Postman.

Commit alle wijzigingen in je lokale repository.

# Opdracht 7

We gaan nu testen met Mocha en Chai toevoegen.

Maak in je projectdirectory een map ‘test’ en daarin een file ‘test.js’. In deze file zet je:

var chai = require('chai');

var chaiHttp = require('chai-http');

var server = require('../server.js');

var should = chai.should();

chai.use(chaiHttp);

describe('API Test', function() {

it('Test GET /api/v2/recipes', function(done) {

chai.request(server)

.get('/api/v2/recipes')

.end(function(err, res) {

res.should.have.status(200);

res.body.should.be.a('array');

done();

});

});

});

In je file server.js waarin je server wordt gedefinieerd zul je een aanpassing moeten doen om te zorgen dat deze testcode jouw server kan opstarten. Onderaan de file server.js komt te staan:

module.exports = server; 🡪 misschien heet jouw app niet ‘server’, aanpassen dus

Run deze eerste geautomatiseerde test vanaf de command line:

mocha

Ga na of de test slaagt, maak aanpassingen tot dat gelukt is.

Voeg nu zelf testen toe voor de volgende routes:

/api/v2/info

/api/v2/recipes/<nummer>

/api/v2/recipes?category=<cat>

Run je tests, pas aan tot ze allemaal werken.

Commit alle wijzigingen in je lokale repository.

# Opdracht 8 (optioneel)

Breid je receptenlijst uit met ingrediënten voor de recepten. Die mag je zelf verzinnen of in het WikiBooks Cookbook opzoeken. Hiervoor zul je de JSON structuur moeten uitbreiden met een item voor ‘ingredients’.

Breid je code uit om ook te kunnen zoeken op ingredients in de receptenlijst, met een query string ‘?ingredients=<ingredient>’. Zoek op welke Javascript functie je daarbij kunt gebruiken om na te gaan of een string voorkomt in een andere string (zit het ingredient uit de query in de lijst van ingrediënten in je recept?).

Commit al je wijzigingen in je lokale repository.

# Opdracht 9 (optioneel)

Deploy je nieuwe recepten-server op Heroku. Daarvoor moet je natuurlijk je lokale repository pushen naar je remote repository op GitHub. Misschien moet je ook een paar files (Procfile, package.json) in je repository aanpassen zodat Heroku weet welke app er gestart moet worden.

# Tot slot

Push je commits naar je remote repository op GitHub.

Demonstreer je recepten-server in het volgende practicum aan de docent.

Succes!