Workshop REST-Assured

Voorbereiding: Zie de readme (https://github.com/RenzoH89/rest-assured-workshop/blob/master/README.md) voor details over de installatie.

Tips

1. Maak gebruik van de mogelijkheid om request en responses te loggen voor meer informatie over de berichten (o.a. log().all)
2. Maak gebruik van de Usage Guide van REST-Assured: https://github.com/rest-assured/rest-assured/wiki/Usage
3. Kom je niet uit een opdracht, vraag hulp of probeer eventueel een andere opdracht.
4. De onderstaande opdrachten zijn ook terug te vinden in het project zelf.

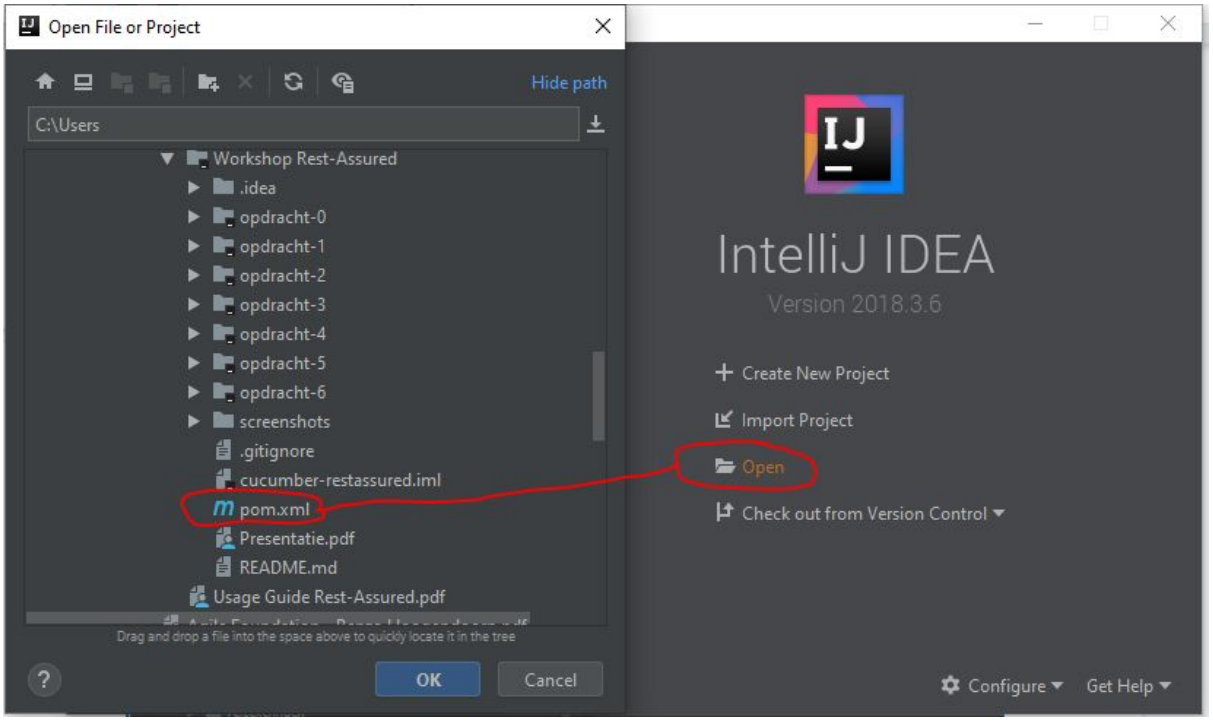
Opdrachten Deel 0: Project uitchecken

Maak bij voorkeur een folder aan genaamd “Workshop REST Assured”. Binnen deze folder voer je het volgende git commando uit:

git clone https://github.com/RenzoH89/rest-assured-workshop.git

Om de eerste opdracht te openen met IntelliJ doorloop je de volgende stappen. Belangrijk is dat het project als Maven project herkend gaat worden in Intellij. Hieronder een mogelijkheid:

1. Kies voor de optie 'Open' als je IntelliJ opent (Welcome to IntelliJ IDEA scherm)
2. Navigeer naar de directory waar het REST-assured project staat
3. Selecteer binnen de folder het pom.xml bestand



4. IntelliJ geeft aan dat de pom.xml een project file is. Kies als optie 'Open as Project'
5. Je bent nu klaar!

Opdrachten Deel 1: Responses valideren

In de volgende opdrachten gaan we de gegevens van een JSON response valideren. De JSON response bevat informatie over een ‘slideshow’. In het project kan je de betreffende URL van het endpoint terug vinden onder de implementatie van de “when” stap.

Module: Opdracht 1

Open binnen de folder opdracht 1 het bestand *‘slideshow.feature’ (opdracht-1/src/test/resourcesfeatures/slideshow.feature).* Om te controleren of alles correct werkt kun je ‘Scenario: 1. The retrieved slideshow should contain 2 slides’ uitvoeren. Deze test zou moeten slagen.

De volgende opdrachten bevinden zich in de ‘slideshow.feature’ file.

Opdracht 1
Bekijk ‘*Scenario: 2. Information about the author of slideshow can be retrieved’*.

Dit scenario werkt op het moment niet.
Maak dit scenario werkend door de regel ‘Then the author of the slideshow is "James"’ te corrigeren

Opdracht 2
Bekijk 'Scenario: 3. The slideshow titles of each slide are returned'.

Implementeer de code achter de stap 'Then the title of the slideshow is "Sample Slide Show"' in de 'SlideshowSteps' klasse. Controleer of de code werkt door het scenario uit te voeren.

Tip: vergeet niet de regel '*throw new cucumber.api.PendingException()*;' te verwijderen uit de desbetreffende stap

Opdracht 3
Bekijk 'Scenario: 3. The slideshow titles of each slide are returned'.

Schrijf zelf een 'then' stap in scenario 3 om de titels van de 2 slides te valideren. Implementeer de code van de stap in de SlideShowSteps klasse.

Opdracht 4
Bekijk 'Scenario: 4. Technical information about the slideshow response should be correct'.

Implementeer de stap 'Then the statuscode 200 is returned'.

Opdracht 5
Bekijk 'Scenario: 4. Technical information about the slideshow response should be correct'.

Schrijf zelf een 'then' stap in scenario 4 om de response header te checken op content-type application/json. Implementeer de code in de SlideShowSteps klasse.

Opdracht 6 (Optioneel)
Er worden nu veel testen uitgevoerd die eigenlijk wel gecombineerd kunnen worden. Voeg de scenario's samen tot er maar 1 scenario over blijft.

Opdracht 7 (Optioneel)
In de bovenstaande opdrachten hebben we constant een JSON response gevalideerd. Maak nu een scenario waarbij we een XML response uit gaan lezen. Gebruik daarvoor een GET op de volgende url: 'https://httpbin.org/xml'

Dit endpoint geeft net als in de bovenstaande opdrachten informatie terug over een 'slideshow'. Aangezien de opbouw niet helemaal hetzelfde is als bij de JSON response is het niet mogelijk om alle eerder geïmplementeerde stappen te hergebruiken. Je kan ervoor kiezen om er losse stappen voor te schrijven, maar wellicht kan je ook nadenken hoe je de stappen wel herbruikbaar kan maken afhankelijk van het type response.

Kies zelf welke velden je wilt valideren.

Opdrachten Deel 2: Requests en Responses

In de volgende opdrachten gaan naast een response valideren ook een simpel request samenstellen. De opdrachten zijn verdeeld over twee feature files: ‘exchangeRate.feature’ en ‘user.feature’.

Module: Opdracht 2

Opdracht 1
Bekijk ‘Scenario: 1. Verify exchange rate for a specific date’ (exchangeRate.feature) binnen de module Opdracht-2 (opdracht-1/src/test/resourcesfeatures/exchangeRate.feature)

Implementeer de code voor de onderstaande stap in de ExchangeRate klasse:

> Given I want to compare the rate of "EUR" against "USD" at "2015-01-02”

Extra informatie:

- De request die we gaan versturen heeft de volgende gegevens nodig:
    - 2 queryParameters
        - Parameter: symbol, Waarde: currencyToCheck parameter uit methode
        - Parameter: base, Waarde :baseCurrency parameter uit methode
    - 1 pathParameter
        - Parameter: date, Waarde: date parameter uit methode
- Sla het request op door de request variabele te gebruiken (request = given()...)

Opdracht 2
Bekijk ‘Scenario: 1. Verify exchange rate for a specific date’ (exchangeRate.feature).

Implementeer de code voor de stap “Then the exchange rate of "EUR" is 0.8303578842 for buying "USD"” in de ExchangeSteps klasse. Controleer hierbij het type valuta en de huidige koers.

Opdracht 3
Bekijk ‘Scenario: 1 A user can be created for which an id and creation date is returned’ (user.feature).

Implementeer de code voor de onderstaande stap in de UserSteps klasse:

> Given a user with name and job is about to be create

Extra informatie:

- De request die we gaan versturen moet bestaan uit een JSON body. Je kan de onderstaande JSON als voorbeeld nemen:
    - {"name": "Bob","job": "Technical Tester"}

- Vergeet in je request niet om de volgende content-type mee te nemen: “application/json”

Opdracht 4
Implementeer de code voor de onderstaande stap in de UserSteps klasse door met de request uit de vorige stap een post te doen naar “https://reqres.in/api/users”. Sla de response vervolgens op in de response variabele.

> When the user is created in the system

Bekijk ‘Scenario: 1 A user can be created for which an id and creation date is returned’ (user.feature).

Opdracht 5
Bekijk ‘Scenario: 1 A user can be created for which an id and creation date is returned’ (user.feature).

Implementeer de code voor de onderstaande stappen in de UserSteps klasse:

> Then an id and creation date for this user is returned
> And the response contains the name and job of the created user

Extra informatie:
- Log de body van response om meer te weten te komen of de velden in de response.
- Maak gebruik van de methode ‘giveToday’ in de UserSteps klasse bij de controle op createdAt waarde.

Opdrachten Deel 3: Authenticatie

In de volgende opdracht gaan we wat testen doen met Basic Authentication. Gebruik de REST assured usage guide voor meer informatie over dit onderwerp: https://github.com/rest-assured/rest-assured/wiki/Usage

Module: Opdracht 3

Opdracht 1
Bekijk ‘Scenario: 1. User can log in when credentials are correct’ (authentication.feature).

Implementeer de code voor de onderstaande stap in de AuthenticationSteps klasse:

> Given my credentials are "user" and "passwd"

Extra informatie:

- Op basis van de parameters dient er een request met basis authenticatie gemaakt te worden.

Opdracht 2
Bekijk 'Scenario: 1. User can log in when credentials are correct' (authentication.feature).

Implementeer de code voor de onderstaande stap in de AuthenticationSteps klasse:

    Then the system grants me access

Controleer daarbij de statuscode van de response en de waarde van het element "authenticated"

Opdracht 3
Bekijk 'Scenario: 2. User cannot log in when credentials are incorrect' (authentication.feature).

De opdracht hier is om het gehele scenario 2 te implementeren. Je kunt hiervoor een aantal stappen uit scenario 1 herbruiken.

Opdrachten Deel 4: Object Mapping

Module: Opdracht 4

Met deze opdrachten gaan we gebruik maken van Object Mapping. Hierbij gaan we JSON files omzetten naar een Object, en andersom. Om je een beetje op weg te helpen is 'Scenario 1' binnen de users.feature file al correct geïmplementeerd.

Opdracht 1
Bekijk 'Scenario: 2. The system returns user details for each user'

In deze opdracht is het de bedoeling of zelf een 'then' stap te schrijven. In deze stap lees je de gebruikers van de response uit op basis van de User klasse en valideer je de waardes id, last_name en avatar uit de response.

Tip: Gebruik de implementatie van scenario 1 om inspiratie op te doen.

Opdracht 2
Bekijk 'Scenario: 2. The system returns user details for each user'

Met de huidige User klasse is het nog niet mogelijk om het element first-name uit te lezen. Pas daarvoor de User klasse aan en valideer vervolgens de response ook op het element first_name uit de response

Opdracht 3

Bekijk ‘Scenario: 3. A user can log in with an email and password’

Voor dit scenario is de stap ‘ Given my credentials are "eve.holt@reqres.in" and "123456"’ nog niet geïmplementeerd. Maak in deze stap een object van Login en gebruik deze in je request.

Opdrachten Deel 5: Reusability

Module: Opdracht 5

Zodra je deze branch hebt uitgecheckt en het project hebt bekeken zul je zien dat er in de ‘stepdefs’ package een aantal nieuwe klasses zijn gemaakt, namelijk: “CommonSteps” en “BaseSteps”.

De “CommonSteps” gaan we gebruiken voor code die je over meerdere feature files en stepdefinitions wilt herbruiken. Je wilt bijv. 1 methode hebben om de statuscode van een response te valideren. Dit gaat niet zomaar, en hiervoor gebruiken we de “StepBaseData” klasse.

Opdracht 1
De methode “theSystemReturnsStatusCode” is op dit moment in meerdere stepdefs geïmplementeerd, en dat werkt niet. Probeer maar eens een scenario te runnen. Schrijf de code om zodat de methode om de statuscode te controleren voor alle scenario’s uit de CommonSteps gebruikt wordt.

Tip: Kijk naar de samenhang van de CommonSteps en de BaseSteps.

Opdracht 2
Het is ook niet altijd handig om losse steps te schrijven voor alle validaties die je wilt doen. In de CommonSteps vind je een methode om de velden uit een Response te valideren, namelijk “theResponseIncludesTheFollowing”.

Vervang in country.feature file de regel “And the official name "Netherlands" and region "Europe" is returned” met de methode uit de CommonSteps. De schrijfwijze hiervoor is als volgt:

> And the response includes the following:
> | path | value |

Doe daarna hetzelfde voor de regel “Then the author of the slideshow is "Yours Truly"” uit de slideshow.feature file.

Opdracht 3
Gebruik de bovenstaande methode om meer velden uit de responses te valideren.

Opdrachten Deel 6: Extract

Module: Opdracht 6

Soms is het nodig om gegevens uit een response te gebruiken voor een aanroep naar een volgende service. In deze opdracht gaan we hiermee aan de slag.

Opdracht 1
Bekijk 'Scenario: 3. Retrieve the author's information based on the first blog post'

Implementeer de 'when' stap 'When I retrieve information about the author of the first blog post'
Zoek op basis van de eerste post uit de response de bijbehorende user met de volgende url: https://jsonplaceholder.typicode.com/users/{idOfUser} waarbij {idOfUser} vervangen dient te worden met het id van de user uit de vorige response.

Opdracht 2 (Optioneel)
Bekijk 'Scenario: 3. Retrieve the author's information based on the first blog post'

Controleer ook de stad waar de gebruiker vandaan komt.

Opdracht 3 (Optioneel)
Maak de eerder aangepaste 'when' step generieker zodat je als parameter mee kan geven van welke post je de gebruiker op wilt halen. Dit mag op basis van id zijn of bijvoorbeeld de titel van een post.