

# WORKSHOP REST-ASSURED

Benodigde software:

- GIT
- IntelliJ IDEA Community Edition (2018.3)
- Java JDK 8

Clone de verschillende branches van het project naar verschillende folders. Zie hiervoor **Opdracht 0** uit de Workshop:

<https://tinyurl.com/workshop-rest>

## WIFI

Naam: Studio-B-5G

Wachtwoord: JP0!ntbv



# REST-ASSURED

Door Renzo Hoogendoorn  
deTesters

*Workshop ondersteuning*  
Richard Duinmaijer  
Koenraad Appelo

# AGENDA

- Wie ben ik
- Wat is een API?
- Wat is REST?
- Waarom API testing?
- Tooling voor testen van API's
- Wat is REST-Assured?
- Voorbeelden en mogelijkheden REST-Assured
- Waarom Cucumber en REST-Assured?
- Introductie Workshop Project

# WIE BEN IK?

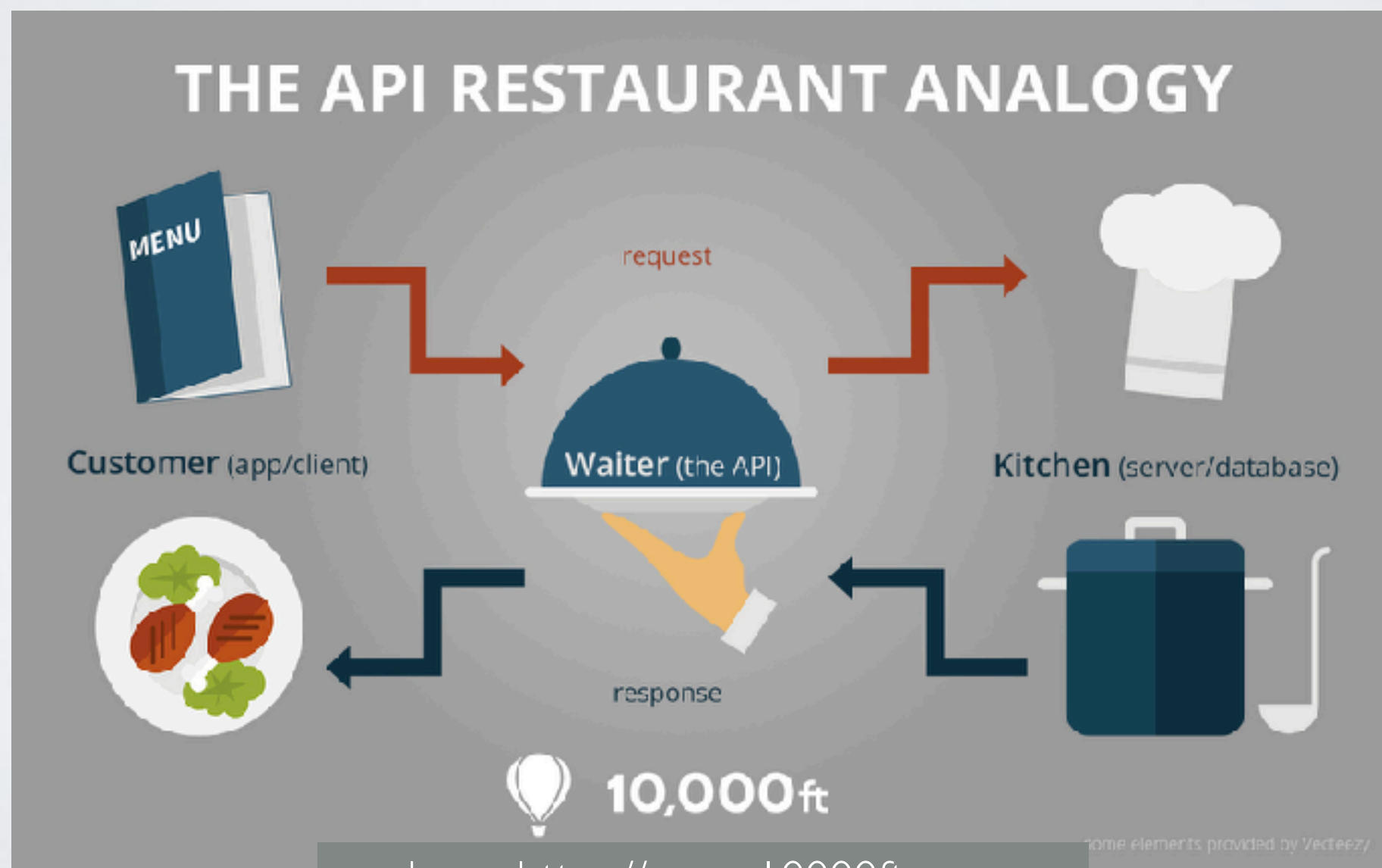
- Renzo Hoogendoorn ([Renzo@detesters.nl](mailto:Renzo@detesters.nl))
- In augustus 2013 begonnen met testen
- Momenteel actief bij de KvK
- Hobbies:



# WAT IS EEN API?

bron: <https://en.wikipedia.org/wiki/>

An application programming interface (API) is a set of subroutine definitions, protocols, and tools for building application software.



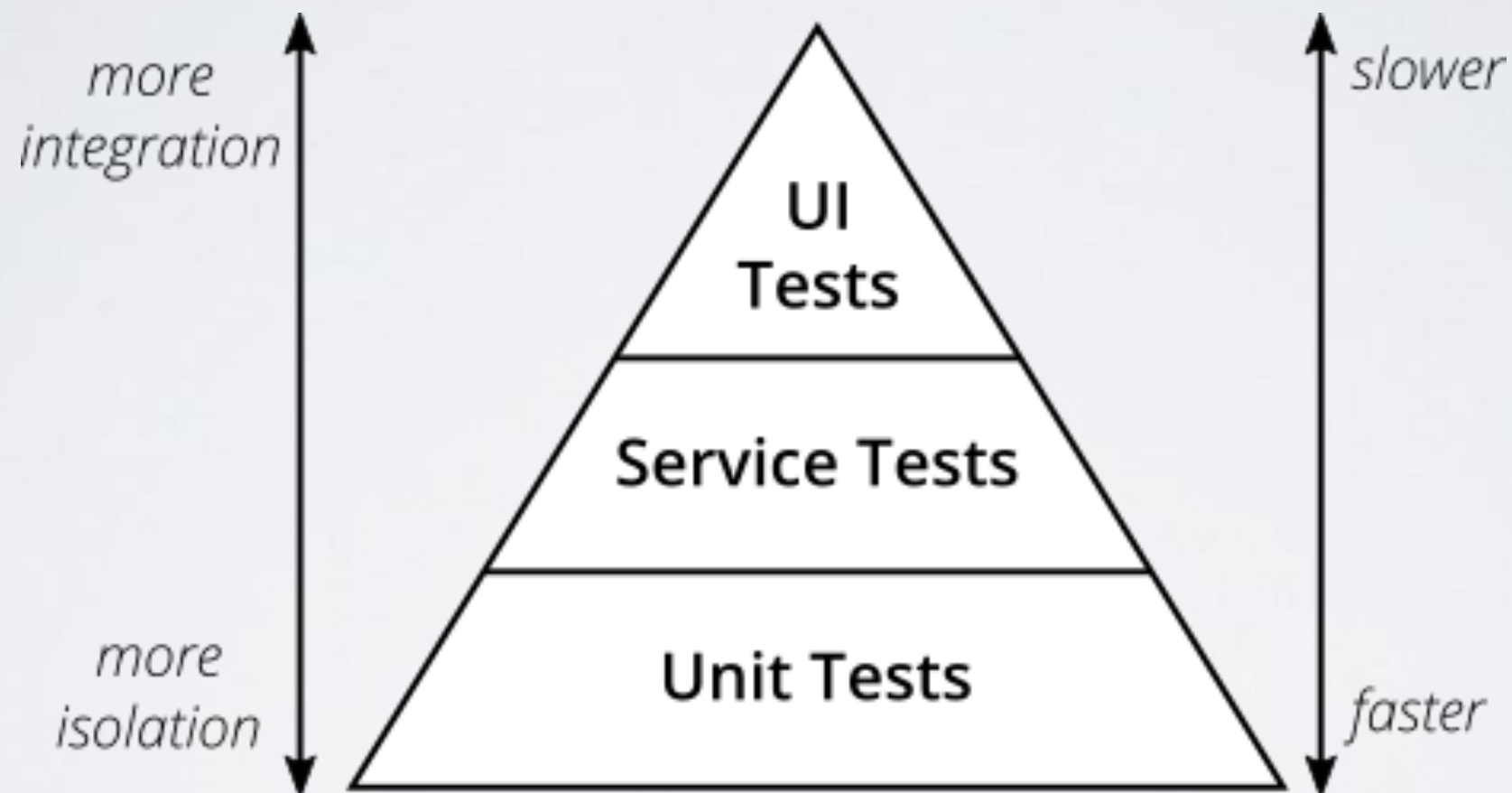
bron: <https://www.10000ft.com>

# WAT IS REST?

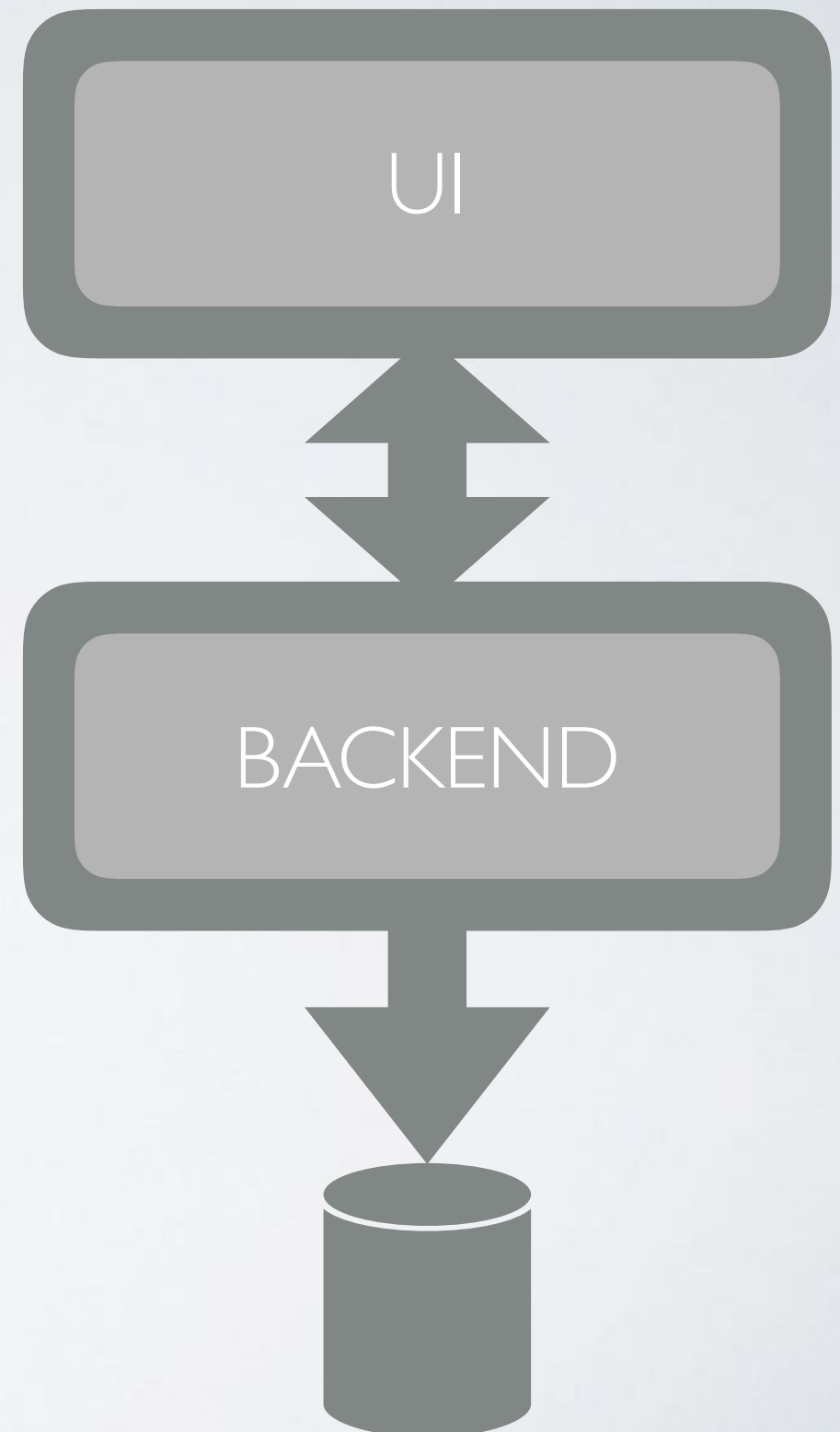
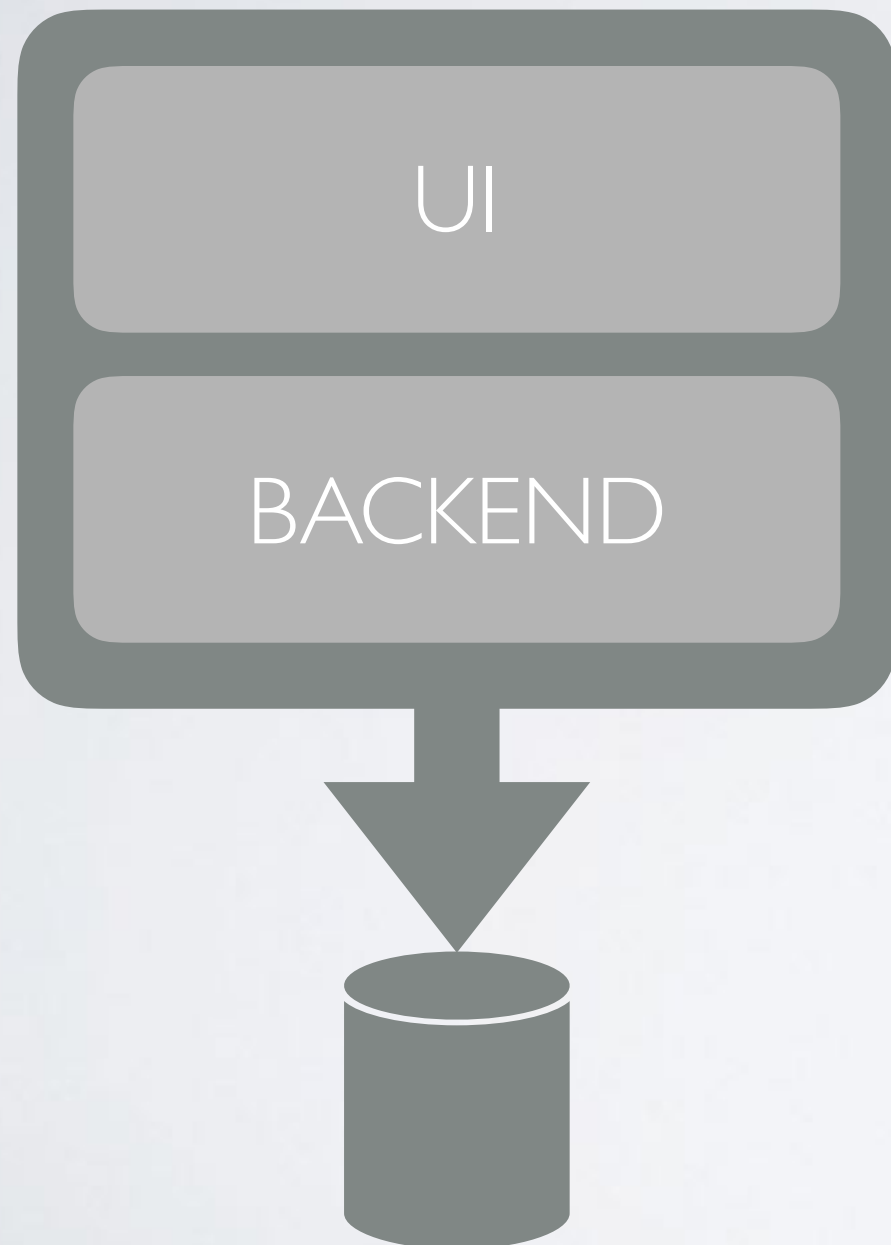
- REpresentational State Transfer
- Endpoints die de operatie beschrijven in combinatie met HTTP methode
- Werkt o.a. met tekst, JSON, XML en HTML
- Loskoppeling client en server
- Werkt ook in je browser



# WAAROM API TESTEN?



# WAAROM API TESTEN?





# TOOLING VOOR TESTEN API'S

- Op zoek naar een geschikte tool
- Er is enorm veel beschikbaar, voor ieder wat wils



# WAAROM REST-ASSURED?

- Onze backend is in Java geschreven. Ontwikkelaars daarmee aan boord
- Door Java ook makkelijk gebruik van WireMock
- Zeer uitgebreid
- Gherkin syntax
- Open Source

# WAT IS REST-ASSURED?

- Een Java library met een goed bruikbare DSL om REST API's testen.
- Ondersteunt POST, GET, PUT, DELETE, OPTIONS, PATCH en HEAD requests (ook custom)
- Response gemakkelijk valideren (JSON, XML)

# HOE EIGEN GEMAAKT?

- Usage Guide (<https://github.com/rest-assured/rest-assured/wiki/Usage>)
- Vallen en opstaan
- Stack Overflow

# VOORBEELD

**Endpoint:** <https://date.nager.at/Api/v2/CountryInfo>

**Query parameter:** CountryCode

**HTTP method:** GET

```
{  
  "officialName": "Netherlands",  
  "region": "Europe"  
}
```

```
@Test  
public void retrieveCountryInformation() {  
    given().  
        queryParams( s: "CountryCode", ...objects: "NL").  
    when().  
        get( s: "https://date.nager.at/Api/v2/CountryInfo").  
    then().  
        statusCode(200).  
        body( s: "officialName", equalTo( operand: "Netherlands"),  
              ...objects: "region", equalTo( operand: "Europe"));  
}
```

# REQUEST SAMENSTELLEN

- Na de Given() kunnen allerlei gegevens gespecificeerd worden (parameters, headers, etc)

# REQUEST SAMENSTELLEN

- Parameters

Los specificeren

```
given().
```

```
    param("param", "value")  
    formparam("param", "value")  
    queryparam("param", "value")
```

Direct in URL

```
request("http://www.url-to-check?param=value")
```



# REQUEST SAMENSTELLEN

- Content Type

```
given().
```

```
contentType("application/json")  
contentType("application/xml")
```

# REQUEST SAMENSTELLEN

- Body

```
given().
```

```
body( "{ \"name\": \"Renzo\" } ")
```

- Werkt ook met bijv. een JSON file:

```
File jsonleirecord = new File(getClass().  
getResource("name.json").getFile());  
given().body(jsonleirecord).contentType(ContentType.JSON)
```

# REQUEST SAMENSTELLEN

- Body op basis van object

```
public class Person {  
    private String name;  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```



```
Person person = new Person();  
person.setName("Renzo");  
given().contentType(ContentType.JSON).body(person)
```



```
{ "name": "Renzo" }
```

# REQUEST SAMENSTELLEN

- Authenticatie
- Cookies
- Headers
- Multi Part

# REQUEST UITVOEREN

- Op basis van HTTP methodes (GET, POST, etc)

`when () .`

```
get("http://www.testurl.nl/1")  
post("http://www.testurl.nl/1")  
request("custom", "http://www.testurl.nl/1")
```

# REQUEST UITVOEREN

- Chaining van methodes

```
given().  
    queryParams("countryCode", "NL").  
when().  
    get("http://www.testurl.nl/1").
```

- Eerst in variabele opslaan (Workshop)

```
RequestSpecification request = given().  
    queryParams("countryCode", countryCode);  
  
request.when().  
    get("http://www.testurl.nl/1")
```

# WAT HEBBEN WE NU?

```
@Test
public void retrieveCountryInformation() {
    given().
        queryParams( s: "CountryCode", ...objects: "NL").
    when().
        get( s: "https://date.nager.at/Api/v2/CountryInfo").
    then().
        statusCode(200).
        body( s: "officialName", equalTo( operand: "Netherlands"),
            ...objects: "region", equalTo( operand: "Europe"));
}
```



# RESPONSE

```
@Test
public void retrieveCountryInformation() {
    given().
        queryParams( s: "CountryCode", ...objects: "NL").
    when().
        get( s: "https://date.nager.at/Api/v2/CountryInfo").
    then().
        statusCode(200).
        body( s: "officialName", equalTo( operand: "Netherlands"),
            ...objects: "region", equalTo( operand: "Europe"));
}
```

# RESPONSE

- Elk request geeft een 'Response' terug. Deze bevat allerlei informatie:

```
Response response = when().get("http://www.test.nl");  
String body = response.getBody();  
String cookie = response.getCookie("cookie1");
```

- Door then() te gebruiken krijg je een Validatable Response terug. Deze kan je gebruiken om direct te valideren

# RESPONSE

- Chaining van methodes

```
when() .  
  get("http://www.test.nl") .  
then() .  
  statusCode(200) ;
```

- Response eerst opslaan (Workshop)

```
Response response = when().get("http://www.test.nl") ;  
response.then().statusCode(200) ;
```

# RESPONSE VALIDEREN

- Body valideren

```
then() .  
  body("page", equalTo(2)) .  
  body("data[0].id", is(3));
```

Hetzelfde:

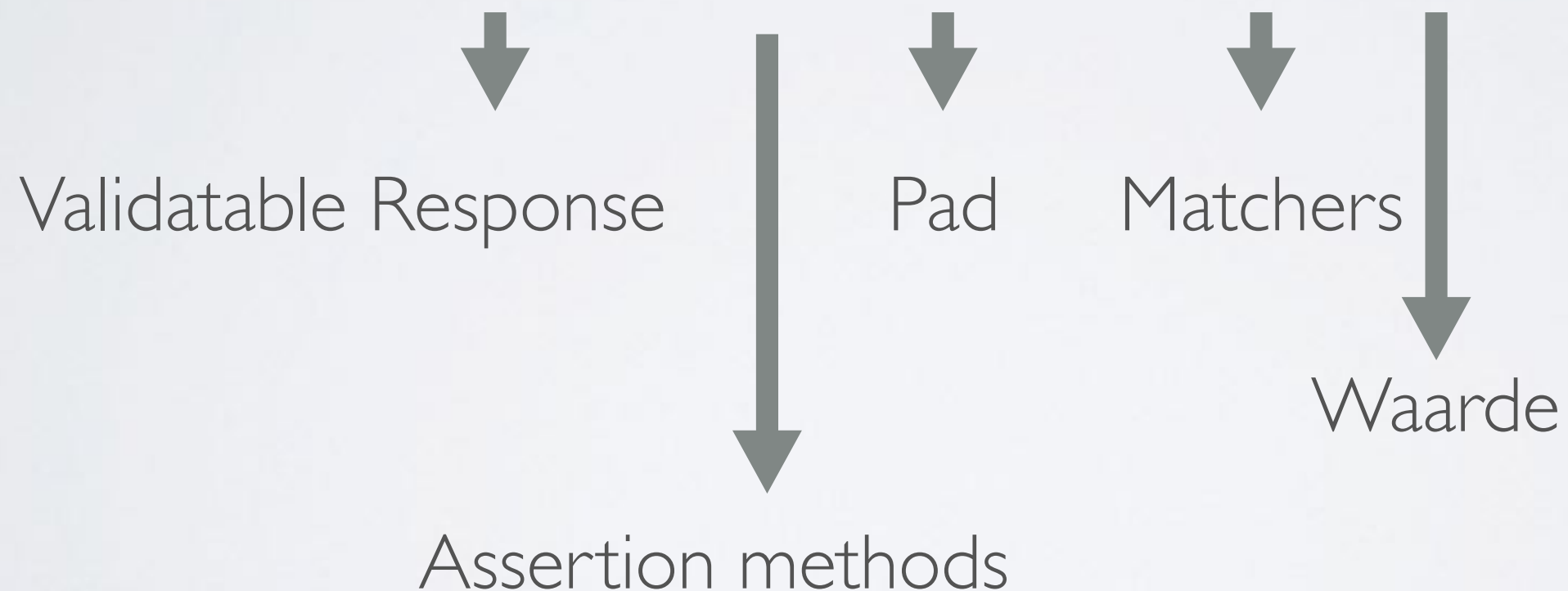
```
then() .  
  assertThat().body("page", equalTo(2)) .  
  "data[0].id", is(3));
```

```
{  
  "page": 2,  
  "per_page": 2,  
  "total": 12,  
  "total_pages": 6,  
  "data": [  
    {  
      "id": 3,  
      "first_name": "Henk",  
      "last_name": "Appel"  
    },  
    {  
      "id": 4,  
      "first_name": "Jos",  
      "last_name": "Peer"  
    }  
  ]  
}
```

# RESPONSE VALIDEREN

- Body valideren

```
then().body("page", equalTo(2))
```



# RESPONSE VALIDEREN

- Body valideren

```
then() .  
    body("data.id", hasItems(3, 4);
```

```
{  
  "page": 2,  
  "per_page": 2,  
  "total": 12,  
  "total_pages": 6,  
  "data": [  
    {  
      "id": 3,  
      "first_name": "Henk",  
      "last_name": "Appel"  
    },  
    {  
      "id": 4,  
      "first_name": "Jos",  
      "last_name": "Peer"  
    }  
  ]  
}
```

# RESPONSE VALIDEREN

- Het valideren van XML is heel vergelijkbaar

```
then() .  
  body("users.page", equalTo(2)) .  
  body("users.data.user[0].id", is(3));
```

Werkt ook met namespaces!

```
<?xml version="1.0" encoding="UTF-8"?>  
<users>  
  <page>2</page>  
  <per_page>3</per_page>  
  <total>12</total>  
  <total_pages>4</total_pages>  
  <data>  
    <user>  
      <id>3</id>  
      <first_name>Henk</first_name>  
      <last_name>Appel</last_name>  
    </user>  
    <user>  
      <id>4</id>  
      <first_name>Jos</first_name>  
      <last_name>Peer</last_name>  
    </user>  
  </data>  
</users>
```



# RESPONSE VALIDEREN

- Kan ook met xPath

```
then() .  
  body(hasXPath("/users/data/user[0]/  
  first_name", is("Henk"))
```

```
<?xml version="1.0" encoding="UTF-8"?>  
<users>  
  <page>2</page>  
  <per_page>3</per_page>  
  <total>12</total>  
  <total_pages>4</total_pages>  
  <data>  
    <user>  
      <id>3</id>  
      <first_name>Henk</first_name>  
      <last_name>Appel</last_name>  
    </user>  
    <user>  
      <id>4</id>  
      <first_name>Jos</first_name>  
      <last_name>Peer</last_name>  
    </user>  
  </data>  
</users>
```


# RESPONSE VALIDEREN

- Body op basis van object

```
{ "name": "Renzo" }
```



```
Person person = when().get("http://...").as(Person.class);  
assertThat(person.getName(), equalTo("Renzo"));
```



```
public class Person {  
    private String name;  
  
    public String getName() {  
        return name;  
    }  
}
```

# RESPONSE VALIDEREN

- Complexe zoekmethode

```
then().body("data.findAll{it.id > 3}.first_name",  
hasItems("Henk", "Huub"));
```

```
then().body("data.id.collect{it.length()}.sum()",  
greaterThan(1));
```

# RESPONSE VALIDEREN

- Direct matchen op XSD
- Namespaces meenemen in de XML check
- Headers, Cookies, etc.
- Extract

# LOGGING

- Requests: Bij problemen is er functionaliteit om makkelijk te debuggen

```
given().log().all()  
body()  
param()  
cookies()
```

```
Request method: GET  
Request URI: https://date.nager.at/Api/v2/CountryInfo?countryCode=NL  
Proxy: <none>  
Request params: <none>  
Query params: countryCode=NL  
Form params: <none>  
Path params: <none>  
Headers: Accept= */*  
Cookies: <none>  
Multiparts: <none>  
Body: <none>
```

# LOGGING

- Voor responses ook mogelijk:

```
then().log().
```

```
all()  
body()  
param()  
cookies()
```

- Maar kan ook enkel loggen bij fouten:

```
then().log().ifError();  
then().log().ifStatusCodeIsEqualTo(500);  
then().log().ifValidationFails().statusCode(200);
```

# RESPONSE TIJD

- Response tijden meten

```
then().time().(lessThan(5L), SECONDS);
```



# WAAROM CUCUMBER

- Leesbaarheid binnen team en naar business
- Herbruikbaarheid van code
- Mooiere rapportage
- Haakt mooi in elkaar (Given, When, Then)

# VERSCHIL

```
@Test
public void retrieveCountryInformation() {
    given().
        queryParams( s: "CountryCode", ...objects: "NL").
    when().
        get( s: "https://date.nager.at/Api/v2/CountryInfo").
    then().
        statusCode(200).
        body( s: "officialName", equalTo( operand: "Netherlands"),
              ...objects: "region", equalTo( operand: "Europe"));
}
```

VS

**Scenario:** 1. System responds with information about the name and region of provided country  
**Given** I want information about the country "NL"  
**When** the information about the country is retrieved  
**Then** the system returns status code 200  
**And** the official name of the country is "Netherlands"  
**And** the region of the country is "Europe"

# IMPLEMENTATIE

```
public class CountryInformationSteps {  
  
    private RequestSpecification request;  
    private Response response;  
  
    @Given("I want information about the country {string}")  
    public void iWantInformationAboutTheCountry(String countryCode) {  
        request = given().queryParams(s: "countryCode", countryCode);  
    }  
  
    @When("the information about the country is retrieved")  
    public void theInformationAboutTheCountryIsRetrieved() {  
        response = request.when().get(s: "https://date.nager.at/Api/v2/CountryInfo");  
    }  
  
    @Then("the system returns status code {int}")  
    public void theSystemReturnsStatusCode(int statusCode) {  
        response.then().statusCode(statusCode);  
    }  
  
    @And("the returned official name of the country is {string}")  
    public void theReturnedOfficialNameOfTheCountryIs(String name) {  
        response.then().body(s: "country", equalTo(name));  
    }  
  
    @And("the returned region of the country is {string}")  
    public void theReturnedRegionOfTheCountryIs(String region) {  
        response.then().body(s: "region", equalTo(region));  
    }  
}
```

# INTRODUCTIE WORKSHOP