Test the REST

Testing RESTful web services using REST Assured

An open source workshop by …

Originally created by Bas Dijkstra - bas@ontestautomation.com - https://www.ontestautomation.com

What are we going to do?

_RESTful APIs

_REST Assured

_Hands-on exercises

Preparation

_Install JDK 1.8 or newer

_Install IntelliJ (or any other IDE)

_Import Maven project into IDE
  _https://github.com/basdijkstra/rest-assured-workshop

(RESTful) APIs are commonly used to exchange data between two parties
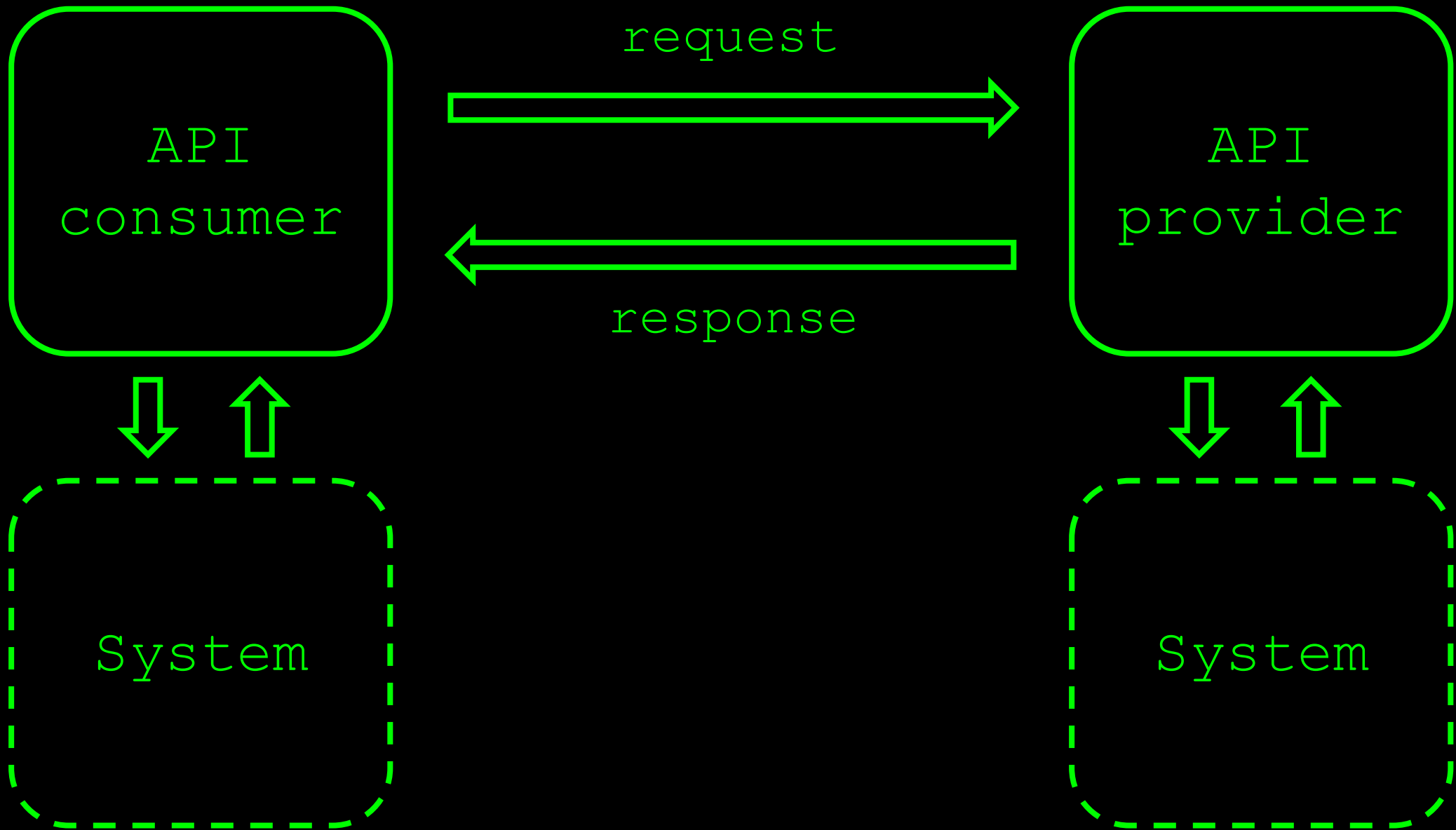
A REST API request

HTTP method | Resource (URI) and parameters

Request headers

Request body

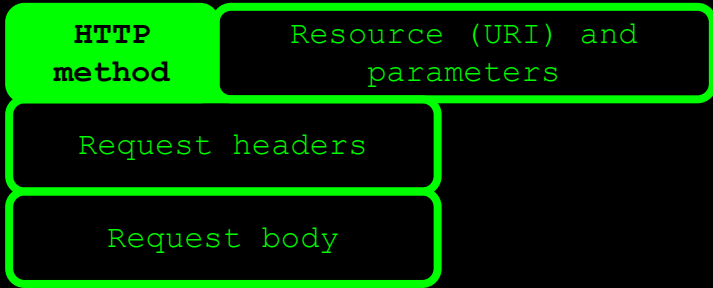# HTTP methods

HTTP method | Resource (URI) and parameters

Request headers

Request body

_GET, POST, PUT, PATCH, DELETE, OPTIONS, …

_CRUD operations on data
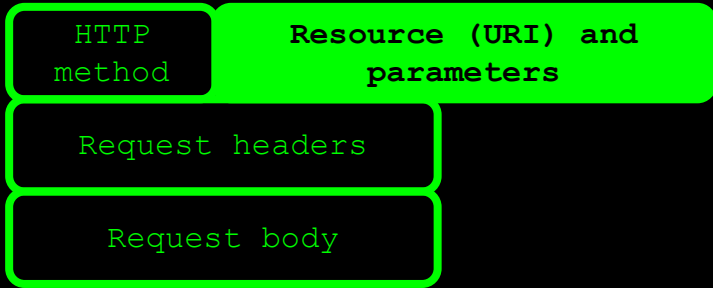 POST   Create
  GET   Read
  PUT / PATCH   Update
  DELETE   Delete
  …   …

_Conventions, not standards!

HTTP
method

**Resource (URI) and
parameters**

Request headers
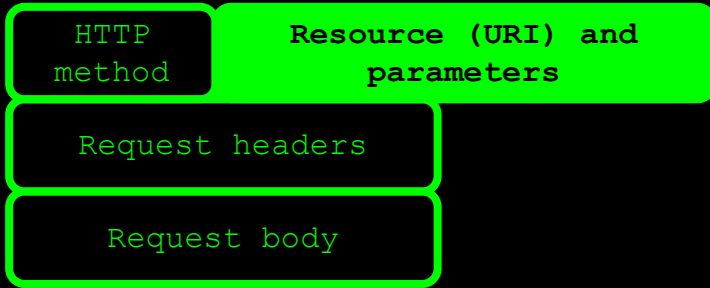
Request body

Resources and parameters

_Uniform Resource Identifier

_Uniquely identifies the resource to operate on

_Can contain parameters
  _Query parameters
  _Path parameters

Resources and parameters
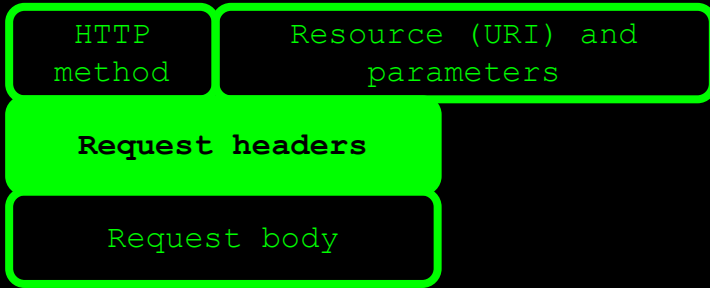
_Path parameters
  _http://api.zippopotam.us/us/90210
  _http://api.zippopotam.us/ca/B2A

_Query parameters
  _http://md5.jsontest.com/?text=testcaseOne
  _http://md5.jsontest.com/?text=testcaseTwo

_There is no official standard!

Request headers

_Key-value pairs

_Can contain metadata about the request body
  _Content-Type (what data format is the request body in?)
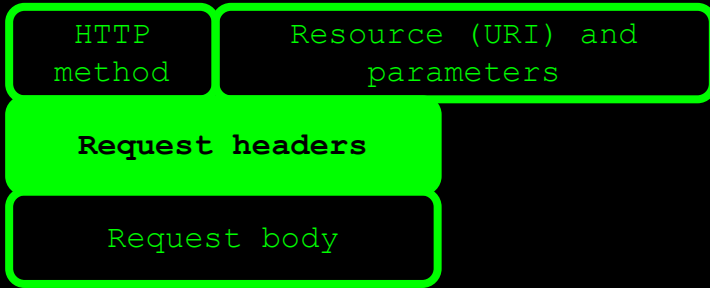 _Accept (what data format would I like the response body to be in?)

  _…

_Can contain session and authorization data
  _Cookies
  _Authorization tokens
  _…

Authorization: Basic

_Username and password sent with every request

_Base64 encoded (not really secure!)

_Ex: username = aladdin and password = opensesame

Authorization: Basic YWxhZGRpbjpvcGVuc2VzYW1l
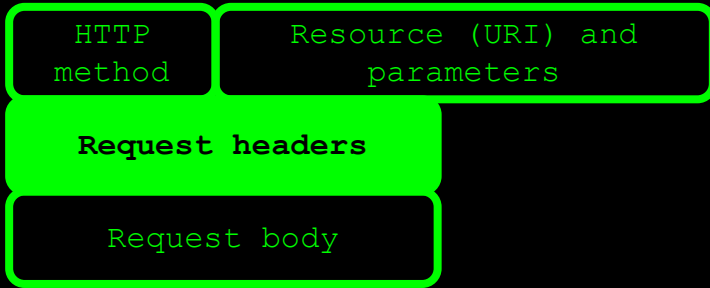
Authorization: Bearer

_Token with limited validity is obtained first

_Token is then sent with all subsequent requests

_Most common mechanism is OAuth(2)

_JWT is a common token format

Authorization: Bearer RsT5OjbzRn430zqMLgV3Ia
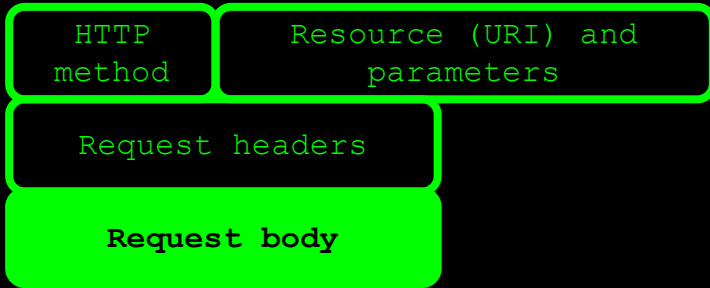
Request body

_Data to be sent to the provider

_REST does not prescribe a specific data format

_Most common:
  _JSON
  _XML
  _Plain text

_Other data formats can be sent using REST, too

A REST API response

HTTP status code

Response headers

Response body

HTTP status code

Response headers

Response body
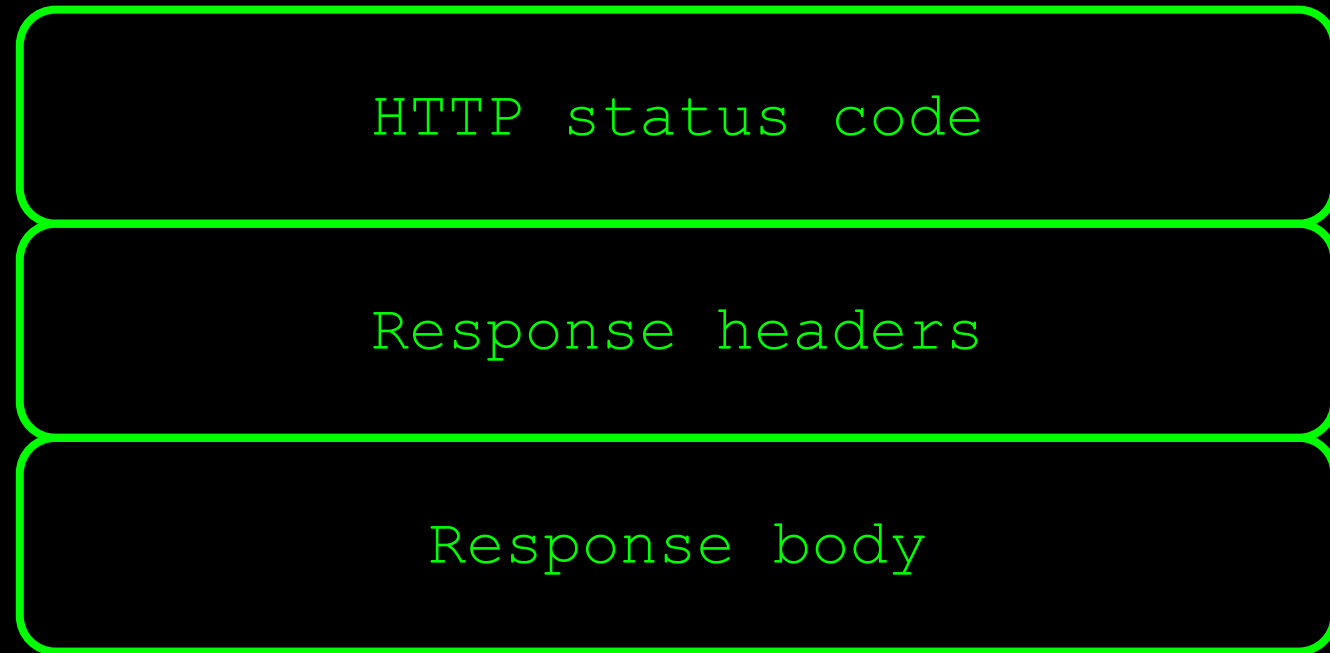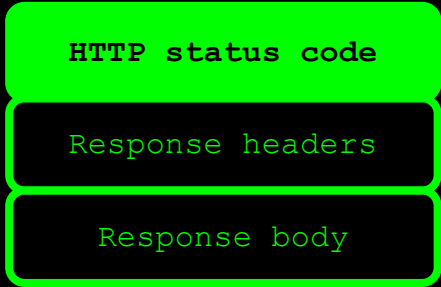
_Indicates result of request processing by provider

_Five different categories

_1XX   Informational   100 Continue

_2XX   Success   200 OK

_3XX   Redirection   301 Moved Permanently

_4XX   Client errors   400 Bad Request

_5XX   Server errors   503 Service Unavailable

Response headers

_Key-value pairs

_Can contain metadata about the response body
  _Content-Type (what data format is the response body in?)
  _Content-Length (how many bytes in the response body?)

_Can contain provider-specific data
  _Caching-related headers
  _Information about the server type

Response body

_Data returned by the provider

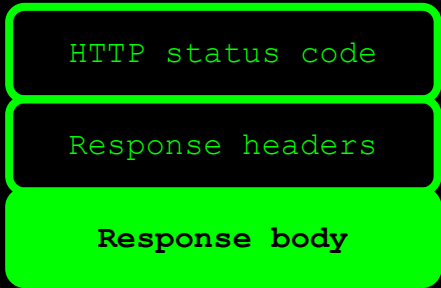_REST does not prescribe a specific data format

_Most common:
  _JSON
  _XML
  _Plain text

_Other data formats can be sent using REST, too

An example

_GET http://ergast.com/api/f1/2018/drivers.json

```
{
- MRData: {
    xmlns: "http://ergast.com/mrd/1.4",
    series: "f1",
    url: "http://ergast.com/api/f1/2018/drivers.json",
    limit: "30",
    offset: "0",
    total: "20",
    - DriverTable: {
        season: "2018",
      - Drivers: [
        - {
            driverId: "alonso",
            permanentNumber: "14",
            code: "ALO",
            url: "http://en.wikipedia.org/wiki/Fernando_Alonso",
            givenName: "Fernando",
            familyName: "Alonso",
            dateOfBirth: "1981-07-29",
            nationality: "Spanish"
        },
        - {
            driverId: "bottas",
            permanentNumber: "77",
            code: "BOT"
```

Headers   Preview   Response   Timing

▼ General

Request URL: http://ergast.com/api/f1/2018/drivers.json
Request Method: GET
Status Code: ● 200 OK
Remote Address: 81.27.85.129:80
Referrer Policy: no-referrer-when-downgrade

▼ Response Headers   view source

Access-Control-Allow-Origin: *
Connection: close
Content-Length: 4494
Content-Type: application/json; charset=utf-8
Date: Tue, 29 Jan 2019 09:39:19 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3

▼ Request Headers   view source

Accept: text/html application/xhtml+xml application/xml

Where are APIs used?

Mobile

Internet of Things
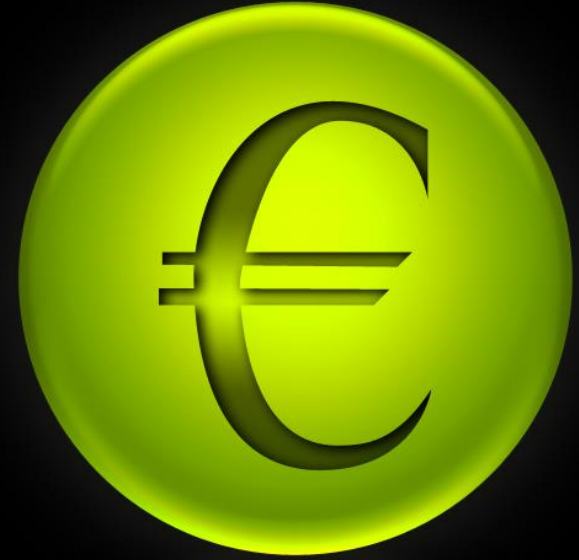
API economy

Where are APIs used?

Web
applications

Microservices
architectures

Why I ♥ testing at the API level

_Tests run much faster than UI-driven tests

_Tests are much more stable than UI-driven tests

_Tests have a broader scope than unit tests

_Business logic is often exposed at the API level

Tools for testing RESTful APIs

_Free / open source
  _ Postman
  _ SoapUI
  _ Code libraries like REST Assured, RestSharp, requests
  _ …

_Commercial
  _ Parasoft SOAtest
  _ SoapUI Pro

  _ …

_Build your own (using HTTP libraries for your language of choice)

REST Assured

_Java DSL for writing tests for RESTful APIs

_Removes a lot of boilerplate code

_Runs on top of common unit testing frameworks
  _JUnit, TestNG

_Developed and maintained by Johan Haleby

Configuring REST Assured

_Download from http://rest-assured.io

_Add as a dependency to your project
  _Maven
  _Gradle

```xml
<dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <version>4.3.1</version>
    <scope>test</scope>
</dependency>
```

REST Assured documentation

_Usage guide
 _https://github.com/rest-assured/rest-assured/wiki/Usage

_Links to other documentation (JavaDoc, getting started, release notes)
 _http://rest-assured.io

A sample test

REST Assured uses JUnit (this could also be TestNG)

```java
@Test
public void getUserData_verifyName_shouldBeLeanneGraham() {

    given().  Make an HTTP GET call to retrieve data from the provider
    when().
        get( s: "http://jsonplaceholder.typicode.com/users/1").  // Do a GET call to the specified resource
    then().
        assertThat()                                              // Assert that the value of the element 'name'
        body( s: "name", equalTo( operand: "Leanne Graham"));    // in the response body equals 'Leanne Graham'
}
```

Perform an assertion on the returned response (here: on the JSON response payload)

REST Assured features

_Support for all HTTP methods (GET, POST, PUT, …)

_Support for BDD / Gherkin (Given/When/Then)

_Use of Hamcrest matchers for checks (*equalTo*)

_Use of Jsonpath/GPath for selecting elements from JSON response

```java
@Test
public void getUserData_verifyName_shouldBeLeanneGraham() {

    given().
    when().
        get( s: "http://jsonplaceholder.typicode.com/users/1").  // Do a GET call to the specified resource
    then().
        assertThat().                                            // Assert that the value of the element 'name'
        body( s: "name", equalTo( operand: "Leanne Graham"));    // in the response body equals 'Leanne Graham'
}
```

About Hamcrest matchers

_Express expectations in natural language

_Examples:

    equalTo(X)   Does the object equal X?

    hasItem("Rome")   Does the collection contain an item "Rome"?

    hasSize(3)   Does the size of the collection equal 3?

    not(equalTo(X))   Inverts matcher equalTo()

_ http://hamcrest.org/JavaHamcrest/javadoc/1.3/org/hamcrest/Matchers.html

About GPath

_JsonPath is a query language for JSON documents
  _REST Assured uses the GPath implementation of JsonPath

_Similar aims and scope as XPath for XML

_Documentation and examples:
  _http://groovy-lang.org/processing-xml.html#_gpath
  _http://groovy.jmiguel.eu/groovy.codehaus.org/GPath.html

GPath example

```json
{
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
        "street": "Kulas Light",
        "suite": "Apt. 556",
        "city": "Gwenborough",
        "zipcode": "92998-3874",
        "geo": {
            "lat": "-37.3159",
            "lng": "81.1496"
        }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org"
```

body("address.geo.lat", equalTo("-37.3159"));

Validating technical response data

_HTTP status code

_Response Content-Type header

_Other headers and their value

_Cookies and their value

—…

```java
@Test
public void getUserData_verifyStatusCodeAndContentType() {

    given().
    when().
        get( s: "http://jsonplaceholder.typicode.com/users/1").
    then().
        assertThat().
        statusCode(200).
    and().
        contentType(ContentType.JSON);
}
```

Logging request data

```java
@Test
public void logAllRequestData() {

    given().
        log().all().
    when().
        get( s: "http://jsonplaceholder.typicode.com/users/1").
    then().
        assertThat().
        body( s: "name", equalTo( operand: "Leanne Graham"));
}
```

log().all() after *given()* logs all request
data to the console

You can also use *log().body()*,
log().headers() as well as other options

Logging request data

```java
@Test
public void logAllRequestData() {

    given().
        log().all().
    when().
        get( s: "http
    then().
        assertThat()
        body( s: "nam
}
```

```
Request method:   GET
Request URI:      http://jsonplaceholder.typicode.com/users/1
Proxy:            <none>
Request params:   <none>
Query params:     <none>
Form params:      <none>
Path params:      <none>
Headers:          Accept=*/*
Cookies:          <none>
Multiparts:       <none>
Body:             <none>
```

Logging response data

```java
@Test
public void logAllResponseData() {

    given().
    when().
        get( s: "http://jsonplaceholder.typicode.com/users/1").
    then().
        log().all().
    and().
        assertThat().
        body( s: "name", equalTo( operand: "Leanne Graham"));
}
```

log().all() after *then()* logs all response data to the console

You can also use *log().body()*, *log().headers()* as well as other options

Logging response data

```java
@Test
public void logAllResponseData() {

    given().
    when().
        get( s: "http://jsonplaceholder.
    then().
        log().all().
    and().
        assertThat().
        body( s: "name", equalTo( operand:
}
```

HTTP/1.1 200 OK
Date: Wed, 28 Oct 2020 16:37:56 GMT
Content-Type: application/json; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: __cfduid=ddc99ce478f9e81d2e127ecfaf8637685160390307{
X-Powered-By: Express
X-Ratelimit-Limit: 1000
X-Ratelimit-Remaining: 993
X-Ratelimit-Reset: 1598842094
Vary: Origin, Accept-Encoding
Access-Control-Allow-Credentials: true
Cache-Control: max-age=43200
Pragma: no-cache
Expires: -1
X-Content-Type-Options: nosniff
Etag: W/"1fd-+2Y3G3w049iSZtw5t1mzSnunngE"
Via: 1.1 vegur
CF-Cache-Status: HIT
Age: 15396
cf-request-id: 0611abd0ce0000e668cd936000000001
Report-To: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\
NEL: {"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 5e9615947bb2e668-LHR
Content-Encoding: gzip

{
    "id": 1

Our API under test

_Zippopotam.us

_Returns location data based
on country and zip code

_http://api.zippopotam.us/

_RESTful API

An example

Path parameters for the country code and zip code

_GET http://api.zippopotam.us/us/90210

```
{
    post code: "90210",
    country: "United States",
    country abbreviation: "US",
    places: [
        {
            place name: "Beverly Hills",
            longitude: "-118.4065",
            state: "California",
            state abbreviation: "CA",
            latitude: "34.0901"
        }
    ]
}
```

The JSON response body

▼ General

Request URL: http://api.zippopotam.us/us/90210
Request Method: GET
Status Code: ● 200 OK
Remote Address: 104.27.136.251:80
Referrer Policy: no-referrer-when-downgrade

▼ Response Headers   view source

Access-Control-Allow-Origin: *
CF-RAY: 4a026ae863a2c797-AMS
Charset: UTF-8
Connection: keep-alive
Content-Encoding: gzip
Content-Type: application/json
Date: Mon, 28 Jan 2019 09:26:28 GMT
Server: cloudflare
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-Cache: hit

Response status code

Response content type

Demo

_API documentation
_How to use the test suite
_Executing your tests
_Reviewing test results

Now it's your turn!

_src > test > java > exercises > RestAssuredExercises1Test.java

_Simple checks
_ _Validating individual elements
_Validating collections and items therein
_Validating technical response properties

_Stubs are predefined
_ _Don't worry about the references to http://localhost
_You only need to write the tests using REST Assured

_Answers are in answers > RestAssuredAnswers1Test.java

_RestAssuredExamples.java contains the examples shown

Parameters in RESTful web services

_Path parameters
  _http://api.zippopotam.us/us/90210
  _http://api.zippopotam.us/ca/B2A

_Query parameters
  _http://md5.jsontest.com/?text=testcaseOne
  _http://md5.jsontest.com/?text=testcaseTwo

_There is no official standard!

Using query parameters

_GET http://md5.jsontest.com/?text=testcase

```java
@Test
public void useQueryParameter() {

                              Define a query parameter and its value
    given().
        queryParam( s: "text",  ...objects: "testcase").
    when().
        get( s: "http://md5.jsontest.com").
    then().
        assertThat().
        body( s: "md5", equalTo( operand: "7489a25fc99976f06fecb807991c61cf"));
}
```

Using path parameters

`_GET http://jsonplaceholder.typicode.com/users/1`

```
@Test
public void usePathParameter() {

    given().       Define a (custom) path parameter name and the parameter value
        pathParam( s: "userId", o: 1).
    when().
        get( s: "http://jsonplaceholder.typicode.com/users/{userId}").
    then().                        Define the location of the path parameter
                                   using the chosen name between {}
        assertThat().
        body( s: "name", equalTo( operand: "Leanne Graham"));
}
```

Exchange data between consumer and provider

GET to retrieve data from provider, POST to send data to provider, …

APIs are all about data

Business logic and calculations often exposed through APIs

Run the same test more than once…

… for different combinations of input and expected output values

Data driven testing

More efficient to do this at the API level…

… as compared to doing this at the UI level

'Feeding' test data to your test

```java
@Test
@DataProvider({
        "1, Leanne Graham",
        "2, Ervin Howell",
        "3, Clementine Bauch"
})
public void checkNameForUser
    (int userId, String expectedUserName) {

    given().
        pathParam( s: "userId", userId).
    when().
        get( s: "http://jsonplaceholder.typicode.com/users/{userId}").
    then().
        assertThat().
        body( s: "name", equalTo(expectedUserName));
}
```

Define test data in the @DataProvider annotation (one record for every iteration, parameters separated by commas)

Use parameters to pass the test data values into the method

Use parameters in the test method where required

Running the data driven test

```java
@Test
@DataProvider({
        "1, Leanne Graham",
        "2, Ervin Howell",
        "3, Clementine Bauch"
})
public void checkNameForUser

    (int userId, String expectedUserName) {

    given().
        pathParam( s: "userId", userId).
    when().
        get( s: "http://jsonplaceholder.typicode.com/users/{userId}").
    then().
        assertThat().
        body( s: "name", equalTo(expectedUserName));
}
```

▼ ✔ RestAssuredExamples (exercises)
   ✔ checkNameForUser[0: 1, Leanne Graham]
   ✔ checkNameForUser[1: 2, Ervin Howell]
   ✔ checkNameForUser[2: 3, Clementine Bauch]

The test method is run
three times, once for
each array ('test case')
in the test data set

Now it's your turn!

src > test > java > exercises > RestAssuredExercises2Test.java

Data driven tests
  Creating a test data object
  Using test data to call the right URI
  Using test data in assertions

Answers are in answers > RestAssuredAnswers2.java

RestAssuredExamples.java contains the examples shown

Authentication

_Securing web services

_Most common authentication schemes:

  _Basic authentication (username / password)

  _OAuth(2)

Basic authentication

```java
@Test
public void useBasicAuthentication() {

    given().

        auth().

        preemptive().

        basic( s: "username", s1: "password")

    when().

        get( s: "https://my.secure/api").

    then().

        assertThat().

        statusCode(200);

}
```

Adding *preemptive()* makes REST
Assured send the credentials
directly, saving us from dealing with
the provider challenging mechanism

OAuth(2)

```java
@Test
public void useOAuthAuthentication() {

    given().

        auth().

        oauth2( s: "myAuthenticationToken").

    when().

        get( s: "https://my.very.secure/api").

    then().

        assertThat().

        statusCode(200);

}
```

The authentication token is typically retrieved prior to running the tests to ensure that a valid token is used

Sharing variables between tests

_Example: uniquely generated IDs

_First call returns a unique ID (e.g. a new user ID)

_Second call needs to use this generated ID

_Since there's no way to predict the ID, we need to
capture and reuse it

Sharing variables between tests

```java
@Test
public void captureAndReuseUserId() {

    String userId =                    The return value can be
                                       stored in a variable…

        given().
        when().
            post( s: "http://my.user.api/user").
        then().
            extract().                 path() takes a GPath
            path( s: "id");            expression to extract
                                       the required value

    given().
        pathParam( s: "userId", userId).
    when().                      … and reused at a later point in time
        get( s: "http://my.user.api/user/{userId}").
    then().
        assertThat().
        statusCode(200);
}
```

RequestSpecifications

_Reuse shared properties shared by many calls

_Base URI

_Port

_Authentication data

_…

Defining and using RequestSpecifications

```java
@Test
public void useRequestSpec() {

    given().
        spec(requestSpec).
    when().
        get( s: "/us/90210.json").
    then().
        assertThat().
        statusCode(200);
}
```

… and use it by calling *spec()* in the *given()* section of your test

```java
private static RequestSpecification requestSpec;

@BeforeClass
public static void createRequestSpec() {


    requestSpec =
        new RequestSpecBuilder().
            setBaseUri("http://api.zippopotam.us").
            setPort(9876).
            build();    Build your RequestSpecification using the Builder pattern…
}
```

Sharing checks between tests

_Example: checking status code and MIME type for all responses

_Another maintenance burden if specified individually for each test

_What if we could specify this once and reuse throughout our tests?

Using a ResponseSpecification

```java
@BeforeClass
public static void createResponseSpec() {

    responseSpec =
            new ResponseSpecBuilder().
                expectStatusCode(200).
                expectContentType(ContentType.JSON).
                build();
}
```

Build your ResponseSpecification using the Builder pattern…

```java
@Test
public void useResponseSpec() {

    given().
    when().
        get("http://jsonplaceholder.typicode.com/users/1").
    then().
        spec(responseSpec).
    and().
        body("name", equalTo("Leanne Graham"));
}
```

… and use it by calling *spec()* in the *then()* section of your test

Now it's your turn!

_src > test > java > exercises > RestAssuredExercises3Test.java

_Apply several options for reuse as shown in the slides

_Answers are in answers > RestAssuredAnswers3Test.java

_RestAssuredExamples.java contains the examples shown

XML support

_So far, we've only used REST Assured on APIs that return JSON

_It works just as well with XML-based APIs

_Identification of response elements uses XmlPath instead of JsonPath

_No need for additional configuration
  _REST Assured uses response content type header value to determine how to process a response body

XmlPath - examples

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<cars>
    <car make="Alfa Romeo" model="Giulia">
        <country>Italy</country>    ⬅
        <year>2016</year>
    </car>
    <car make="Aston Martin" model="DB11">
        <country>UK</country>
        <year>1949</year>
    </car>
    <car make="Toyota" model="Auris">
        <country>Japan</country>
        <year>2012</year>
    </car>
</cars>
```

Check country for the
first car in the list

```java
@Test
public void checkCountryForFirstCar() {

    given().
    when().
        get( S: "http://path.to/cars/xml").
    then().
        assertThat().
        body( S: "cars.car[0].country", equalTo( operand: "Italy"));
}
```

XmlPath - examples

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<cars>
    <car make="Alfa Romeo" model="Giulia">
        <country>Italy</country>
        <year>2016</year>
    </car>
    <car make="Aston Martin" model="DB11">
        <country>UK</country>
        <year>1949</year>
    </car>
    <car make="Toyota" model="Auris">
        <country>Japan</country>
        <year>2012</year>  ⟵
    </car>
</cars>
```

Check year for the
last car in the list

```java
@Test
public void checkYearForLastCar() {

    given().
    when().
        get( s: "http://path.to/cars/xml").
    then().
        assertThat().
        body( s: "cars.car[-1].year", equalTo( operand: "2012"));
}
```

XmlPath - examples

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<cars>
    <car make="Alfa Romeo" model="Giulia">
        <country>Italy</country>
        <year>2016</year>
    </car>
    <car make="Aston Martin" model="DB11">    ←────────
        <country>UK</country>
        <year>1949</year>
    </car>
    <car make="Toyota" model="Auris">
        <country>Japan</country>
        <year>2012</year>
    </car>
</cars>
```

Check model for the
second car in the list

(use an @ to refer to
an XML attribute)

```java
@Test
public void checkModelForSecondCar() {

    given().
    when().
        get( s: "http://path.to/cars/xml").
    then().
        assertThat().
        body( s: "cars.car[1].@model", equalTo( operand: "DB11"));
}
```

XmlPath - examples

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<cars>
    <car make="Alfa Romeo" model="Giulia">
        <country>Italy</country>
        <year>2016</year>
    </car>
    <car make="Aston Martin" model="DB11">
        <country>UK</country>
        <year>1949</year>
    </car>
    <car make="Toyota" model="Auris">
        <country>Japan</country>
        <year>2012</year>
    </car>
</cars>
```

Check there's one car from Japan in the list

findAll is a filter operation

```java
@Test
public void checkTheListContainsOneJapaneseCar() {

    given().
    when().
        get( s: "http://path.to/cars/xml").
    then().
        assertThat().
        body( "cars.car.findAll{it.country=='Japan'}.size()", equalTo( operand: 1));
}
```

XmlPath - examples

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<cars>
    <car make="Alfa Romeo" model="Giulia">
        <country>Italy</country>
        <year>2016</year>
    </car>
    <car make="Aston Martin" model="DB11">
        <country>UK</country>
        <year>1949</year>
    </car>
    <car make="Toyota" model="Auris">
        <country>Japan</country>
        <year>2012</year>
    </car>
</cars>
```

Check that two cars have a make starting with 'A'

grep takes a regular expression to search in a list of values

```java
@Test
public void checkTheListContainsTwoCarsWhoseMakeStartsWithAnA() {

    given().
    when().
        get( s: "http://path.to/cars/xml").
    then().
        assertThat().
        body( s: "cars.car.@make.grep(~/A.*/).size()", equalTo( operand: 2));
}
```

Now it's your turn!

src > test > java > exercises > RestAssuredExercises4Test.java

Communicating with an API returning an XML document

Use XmlPath to select the right nodes

Use filters, in, grep() where needed

Answers are in answers > RestAssuredAnswers4Test.java

RestAssuredExamplesXml.java contains the examples shown

(De-)serialization of POJOs

_REST Assured is able to convert POJO instances
directly to XML or JSON (and back)

_Useful when dealing with test data objects
  _Creating request body payloads
  _Processing response body payloads

_Requires additional libraries on the classpath
  _Jackson or Gson for JSON
  _JAXB for XML

```xml
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.9.8</version>
    <scope>test</scope>
</dependency>
```

Example: serialization

_POJO representing an address

```java
public class Address {

    private String street;
    private int houseNumber;
    private int zipCode;
    private String city;

    public Address(String street, int houseNumber, int zipCode, String city) {

        this.street = street;
        this.houseNumber = houseNumber;
        this.zipCode = zipCode;
        this.city = city;

    }
```

Example: serialization

```java
@Test
public void serializeAddressToJson() {

    Address myAddress = new Address( street: "My street", houseNumber: 1, zipCode: 1234, city: "Amsterdam");

    given().
        body(myAddress).          Pass the object as a request body using body()…
    when().
        post( s: "http://localhost:9876/address").
    then().
        assertThat().
        statusCode(200);
}
```

… and REST Assured will serialize it to JSON using Jackson
(which means you can customize the field names if required)

```
Body:
{"street":"My street","houseNumber":1,"zipCode":1234,"city":"Amsterdam"}
```

Example: deserialization

```java
@Test
public void deserializeJsonToAddress() {

    Address myAddress = … store the deserialized response payload
                           in an object of that type…

        given().
        when().
            get( s: "http://localhost:9876/address").
            as(Address.class);    Specify the object type to deserialize to
                                   using as()…

    Assert.assertEquals( expected: "Amsterdam", myAddress.getCity());
}
```

… store the deserialized response payload in an object of that type…

Specify the object type to deserialize to using *as()*…

… and then use it in the remainder of your test method as required

Now it's your turn!

_src > test > java > exercises > RestAssuredExercises5Test.java

_Practice (de-)serialization for yourself

_You don't need to create or adapt the Car POJO

_Answers are in answers > RestAssuredAnswers5Test.java

_RestAssuredExamples.java contains the examples shown

Now it's your turn!

_src > test > java > exercises > RestAssuredExercises6Test.java

_Capstone assignment

_Combines several concepts we have seen throughout this
workshop
  _Extracting values from responses
  _Deserialization
  _Using filters
  _Parameterization, assertions, …

_Answers are in answers > RestAssuredAnswers6Test.java

https://testautomationu.applitools.com/automating-your-api-tests-with-rest-assured/

Contact

_Email: bas@ontestautomation.com

_Website: https://www.ontestautomation.com/training

_LinkedIn: https://www.linkedin.com/in/basdijkstra