Introduction to API Testing

Lesson 4:Introduction to REST Assured

Capgemini

Lesson Objectives

- What is REST Assured
- WHAT A REST ASSURED GIVES
- REST Assured Setup
- Getting started with REST Assured – Adding Dependencies
- REST API Methods
  - GET
  - POST
  - PUT
  - PATCH
- Summary of HTTP Methods for RESTful APIs
- Writing Test
- HTTP GET Request Example
- HTTP POST Request Example
- Demo
- Summary

What is REST Assured

- REST Assured can be used to test XML as well as JSON based web services.
- REST Assured can be integrated with JUnit and TestNG frameworks for writing test cases for our application.
- REST Assured supports POST, GET, PUT, DELETE, OPTIONS, PATCH, and HEAD requests and can be used to validate and verify the response of these requests.
- Rest Assured has methods to fetch data from almost every part of the request and response no matter how complex the JSON structures are.
- Rest Assured is a open source with a lot of additional methods and libraries being added has made it a great choice for API automation.

What is REST Assured Continue…

- Testing and validating REST services in Java is harder than in dynamic languages such as Ruby and Groovy. REST Assured brings the simplicity of using these languages into the Java domain.
- REST Assured is a simple Java library for testing of REST services.
- Integrates seamlessly with existing Java based testing frameworks
  - Junit, TestNG
  - Selenium Webdriver
- Rest Assured features
  - Supports for HTTP methods
  - Supports for BDD/Gherkin(Given, When, Then)
  - Use of Hamcrest matches for checks (equalTo)
  - Use of Gpath for selecting element from JSON response.

WHAT A REST ASSURED GIVES

- It gives various methods to
  - format the HTTP request
  - send it over a network
  - validating HTTP status code of an request
  - Finally validating response data.
- It also handles the various authentication mechanism for the REST Requests
- Basic Authentication(with username and password)
  - OAuth
  - OAuth 2.0
  - Certificates
  - Digest
  - Form based authentication

REST Assured Setup

- Step 1) Install Java.
- Step 2) Download an IDE to begin: eclipse
- Step 3) InstallMaven and set up your eclipse.

Getting started with REST Assured – Adding Dependencies

We need to update pom.xml with the below mentioned dependencies

- RestAssured
- TestNG
- apache poi
- Json-simple

REST API Methods

- RESTful APIs enable you to develop any kind of web application having all possible CRUD (create, retrieve, update, delete) operations. REST guidelines suggest you to use specific HTTP method on specific type of call made to server.
- Commonly used HTTP Methods are:
  - HTTP GET
  - HTTP POST
  - HTTP PUT
  - HTTP DELETE
  - HTTP PATCH

REST API Methods - HTTP GET

- GET requests is used to retrieve resource representation/information only – and not to modify it in any way.
- Get requests are said to be safe methods.
- GET APIs should be idempotent.
- Status Code on executing GET request:
    - If the resource is found on the server then it must return HTTP response code 200 (OK) along with response body
    - In case resource is NOT found on server then it must return HTTP response code 404
    - if it is determined that GET request itself is not correctly formed then server will return HTTP response code 400

REST API Methods - HTTP POST

- Use POST APIs **to create new subordinate resources**, e.g. a file is subordinate to a directory containing it or a row is subordinate to a database table.
- POST methods are used to create a new resource into the collection of resources.
- Status Code status on executing POST
  - if a resource has been created on the origin server, the response SHOULD be HTTP response code 201 (Created)
  - Many times, the action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either HTTP response code 200 (OK) or 204 (No Content) is the appropriate response status.
- POST is neither safe nor idempotent and invoking two identical POST requests will result in two different resources containing the same information (except resource ids).

REST API Methods - HTTP PUT

- Use PUT APIs primarily **to update existing resource** (if resource does not exist then API may decide to create a new resource or not).
- Status code on executing PUT
  - If a new resource has been created by the PUT API, the origin server MUST inform the user agent via the HTTP response code 201.
  - if an existing resource is modified, either the 200 (OK) or 204 (No Content) response codes SHOULD be sent to indicate successful completion of the request.
  - If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are **not cacheable**.

REST API Methods - HTTP PATCH

- HTTP PATCH requests are **to make partial update on a resource**. If you see PUT requests also modify a resource entity so to make more clear.
- PATCH method is the correct choice for partially updating an existing resource and PUT should only be used if you’re replacing a resource in it’s entirety.
- Support for PATCH in browsers, servers and web application frameworks is not universal. IE8, PHP, Tomcat, django, and lots of other software has missing or broken support for it.
- Request payload of PATCH request in not straightforward as it is for PUT request. e.g HTTP GET /users/1

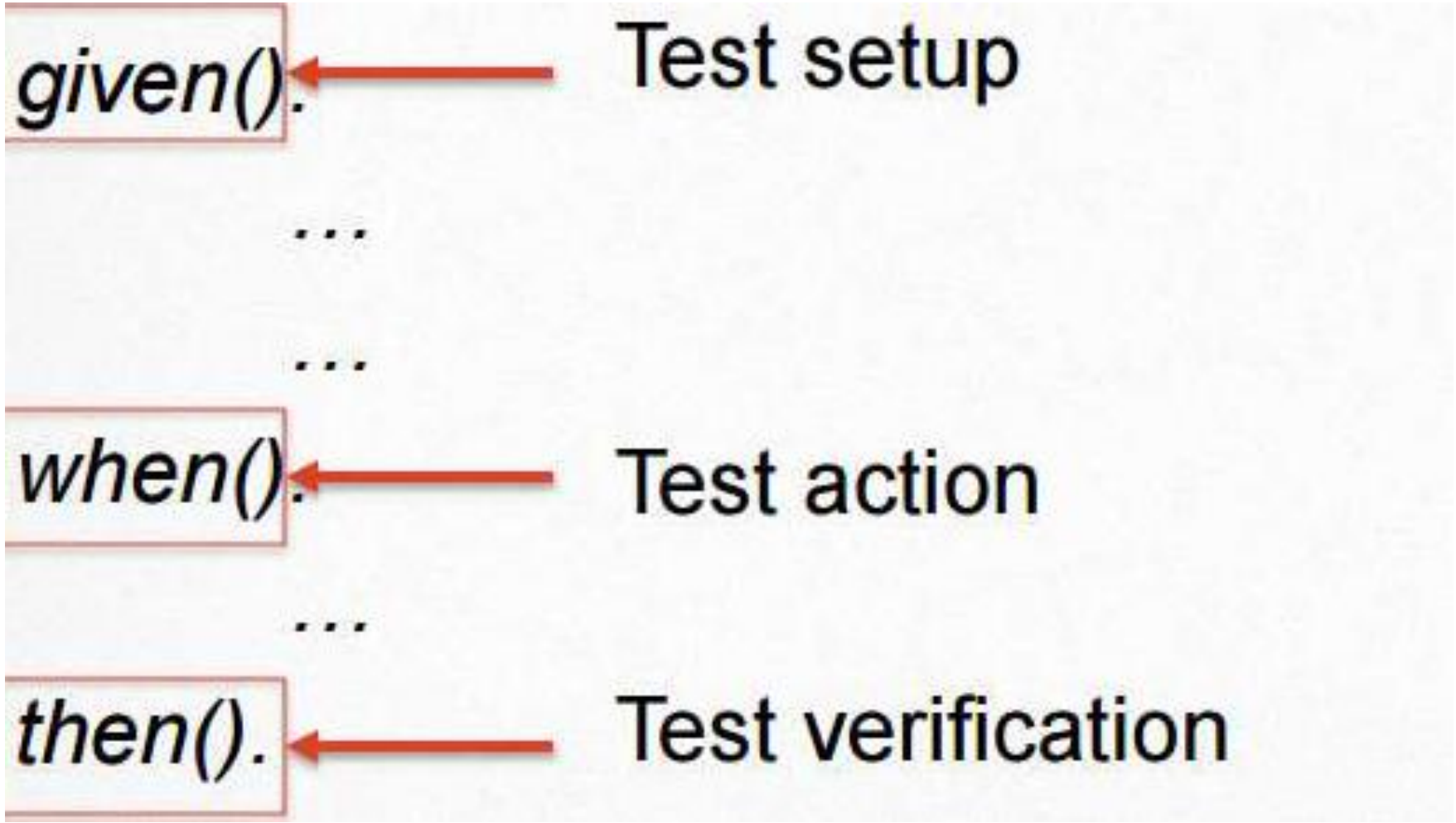
Summary of HTTP Methods for RESTful APIs

HTTP METHOD	CRUD	ENTIRE COLLECTION (E.G. /USERS)	SPECIFIC ITEM (E.G. /USERS/123)
POST	Create	201 (Created), 'Location' header with link to /users/{id} containing new ID.	Avoid using POST on single resource
GET	Read	200 (OK), list of users. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single user. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	404 (Not Found), unless you want to update every resource in the entire collection of resource.	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid.

Summary of HTTP Methods for RESTful APIs

PATCH	Partial Update/Modify	404 (Not Found), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid.
DELETE	Delete	404 (Not Found), unless you want to delete the whole collection — use with caution.	200 (OK). 404 (Not Found), if ID not found or invalid.

Writing Test



HTTP GET Request Example

- request = given()

```
.contentType("application/json“)
.auth()
.basic(authData.getUserName(),authData.getPassword())
.when()
.get(url);
```

HTTP POST Request Example

```
@Test
    public void submitForm() {
        RestAssured.baseURI = "https://www.example.com";
         given().urlEncodingEnabled(true)
            .param("username", "user@site.com")
           .param("password", "Pas54321")
          .header("Accept", ContentType.JSON.getAcceptHeader())
           .post("/login")
            .then().statusCode(200);
    }
}
```

Demo

- Entire Work Flow using Rest Assured Demo

Summary

- In this lesson, you have learnt:
  - What is REST Assured
  - WHAT A REST ASSURED GIVES
  - REST Assured Setup
  - Getting started with REST Assured – Adding Dependencies
  - REST API Methods
  - Summary of HTTP Methods for RESTful APIs
  - Writing Test
  - HTTP GET Request Example
  - HTTP POST Request Example