**Rest Assured Automation Learning**

**Status Code – description**

**Reference** <https://restfulapi.net/http-status-codes/>

**1×× Informational**

100 Continue  
101 Switching Protocols  
102 Processing

**2×× Success**

**200 OK**  
**201 Created**  
**202 Accepted**  
203 Non-authoritative Information  
**204 No Content**  
205 Reset Content  
206 Partial Content  
207 Multi-Status  
208 Already Reported  
226 IM Used

**3×× Redirection**

300 Multiple Choices  
301 Moved Permanently  
302 Found  
303 See Other  
304 Not Modified  
305 Use Proxy  
307 Temporary Redirect  
308 Permanent Redirect

**4×× Client Error**

**400 Bad Request  
401 Unauthorized**  
402 Payment Required  
**403 Forbidden**  
**404 Not Found**  
405 Method Not Allowed  
406 Not Acceptable  
407 Proxy Authentication Required  
**408 Request Timeout**  
409 Conflict  
410 Gone  
411 Length Required  
412 Precondition Failed  
413 Payload Too Large  
414 Request-URI Too Long  
415 Unsupported Media Type  
416 Requested Range Not Satisfiable  
417 Expectation Failed  
418 I’m a teapot  
421 Misdirected Request  
422 Unprocessable Entity  
423 Locked  
424 Failed Dependency  
426 Upgrade Required  
428 Precondition Required  
429 Too Many Requests  
431 Request Header Fields Too Large  
**444 Connection Closed Without Response**  
451 Unavailable For Legal Reasons  
**499 Client Closed Request**

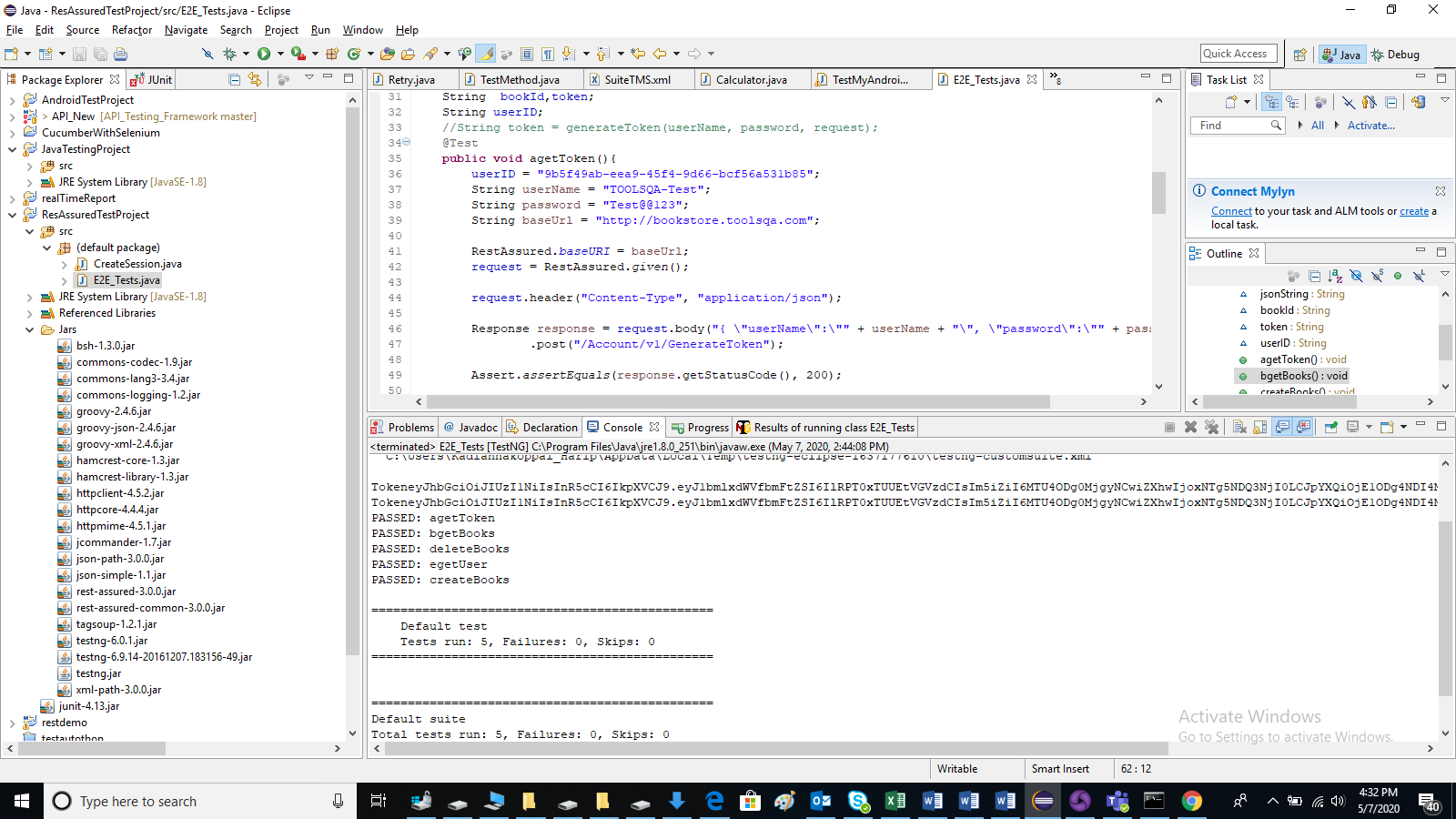
**5×× Server Error**

**500 Internal Server Error**  
501 Not Implemented  
**502 Bad Gateway**  
**503 Service Unavailable**  
**504 Gateway Timeout**  
505 HTTP Version Not Supported  
506 Variant Also Negotiates  
507 Insufficient Storage  
508 Loop Detected  
510 Not Extended  
511 Network Authentication Required  
**599 Network Connect Timeout Error**

**Rest Assured Framework development**

**Reference -** <https://www.toolsqa.com/rest-assured/rest-api-end-to-end-test/>

**Used Eclipse, Java, TestNg, and required set of Jar files including RestAssured jar(All jars are listed in my eclipse workspace)**



**Working API Tests ‘**E2E\_Tests**’**

**import** java.util.List;

**import** java.util.Map;

//import org.junit.Assert;

**import** org.testng.Assert;

**import** org.testng.annotations.BeforeClass;

**import** org.testng.annotations.Test;

**import** io.restassured.RestAssured;

**import** io.restassured.path.json.JsonPath;

**import** io.restassured.response.Response;

**import** io.restassured.specification.RequestSpecification;

**public** **class** E2E\_Tests {

@BeforeClass

**public** **void** createSession(){

}

//Step - 1

//Test will start from generating Token for Authorization

Response response;

RequestSpecification request;

String jsonString;

String bookId,token;

String userID;

//String token = generateToken(userName, password, request);

@Test

**public** **void** agetToken(){

userID = "9b5f49ab-eea9-45f4-9d66-bcf56a531b85";

String userName = "TOOLSQA-Test";

String password = "Test@@123";

String baseUrl = "http://bookstore.toolsqa.com";

RestAssured.*baseURI* = baseUrl;

request = RestAssured.*given*();

request.header("Content-Type", "application/json");

Response response = request.body("{ \"userName\":\"" + userName + "\", \"password\":\"" + password + "\"}")

.post("/Account/v1/GenerateToken");

Assert.*assertEquals*(response.getStatusCode(), 200);

String jsonString = response.asString();

Assert.*assertTrue*(jsonString.contains("token"));

//This token will be used in later requests

token = JsonPath.*from*(jsonString).get("token");

System.***out***.println("Token"+token);

}

//Step - 2

// Get Books - No Auth is required for this.

@Test()

**public** **void** bgetBooks(){

response = request.get("/BookStore/v1/Books");

Assert.*assertEquals*(response.getStatusCode(), 200);

jsonString = response.asString();

List<Map<String, String>> books = JsonPath.*from*(jsonString).get("books");

Assert.*assertTrue*(books.size() > 0);

//This bookId will be used in later requests, to add the book with respective isbn

bookId = books.get(0).get("isbn");

}

//Step - 3

// Add a book - with Auth

//The token we had saved in the variable before from response in Step 1,

//we will be passing in the headers for each of the succeeding request

@Test(priority=3)

**public** **void** createBooks(){

System.***out***.println("Token"+token);

request.header("Authorization", "Bearer " + token)

.header("Content-Type", "application/json");

response = request.body("{ \"userId\": \"" + userID + "\", " +

"\"collectionOfIsbns\": [ { \"isbn\": \"" + bookId + "\" } ]}")

.post("/BookStore/v1/Books");

Assert.*assertEquals*( 201, response.getStatusCode());

}

//Step - 4

// Delete a book - with Auth

@Test

**public** **void** deleteBooks(){

request.header("Authorization", "Bearer " + token)

.header("Content-Type", "application/json");

response = request.body("{ \"isbn\": \"" + bookId + "\", \"userId\": \"" + userID + "\"}")

.delete("/BookStore/v1/Book");

Assert.*assertEquals*(204, response.getStatusCode());

}

//Step - 5

// Get User

@Test

**public** **void** egetUser(){

request.header("Authorization", "Bearer " + token)

.header("Content-Type", "application/json");

response = request.get("/Account/v1/User/" + userID);

Assert.*assertEquals*(200, response.getStatusCode());

jsonString = response.asString();

List<Map<String, String>> booksOfUser = JsonPath.*from*(jsonString).get("books");

Assert.*assertEquals*(4, booksOfUser.size());

}

}

**Just right click on this file and run as TestNG - > all tests will get pass.**

**Interview Preparation RestFull webservice testing**

**Q #4) Enlist features of RESTful web services.**

**Answer:**Every RESTful web services should have the following features and characteristics that are enlisted below:

* Based on the Client-Server representation.
* Use of HTTP protocol for performing functions like fetching data from the web service, retrieving resources, execution of any query, etc.
* The communication between the server and client is performed through the medium known as ‘messaging’.
* Addressing of resources available on the server through URIs.
* Based on the concept of statelessness where every client request and the response is independent of the other with complete assurance of providing required information.
* Uses the concept of caching.
* Works on the Uniform interface.

**Q #5) Explain messaging technique.**

**Answer:**Messages are the mode of exchanging data for any type of communication to take place. In the same way, HTTP protocol plays the role of message communication between the client and server through HTTP Request and Response methods. HTTP request is sent by the client who contains information about the data and in turn, receives HTTP Response from the server.

Messages are the collection of information about the data i.e. Metadata.

**Q #6) What are the core components of the HTTP request and HTTP response?**

**Answer:**The core components under HTTP Request are:

* **Verb:** Includes methods like GET, PUT, POST, etc.
* Uniform Resource Identifier for identifying the resources available on the server.
* HTTP Version for specifying the HTTP version.
* HTTP Request header for containing the information about the data.
* HTTP Request body that contains the representation of the resources in use.

**Q #10) What is a ‘Resource’?**

‘Resource’ is defined as an object

**Some most common Resources are enlisted below:**

* JSON
* YAML
* XML
* HTML
* The resource representation format should be easily understood by the client and server.
* The representation should be complete regardless of its format structure, which may be complex or simple.
* **Q #13) What is Caching?**
* **Answer:**Caching is the process in which server response is stored so that a cached copy can be used when required and there is no need for generating the same response again. This process not only reduces the server load but in turn increase the scalability and performance of the server. Only the client is able to cache the response and that too for a limited period of time.
* **Q #16) What is Payload?**
* **Answer:**The request data which is present in the body part of every HTTP message is referred to as ‘Payload’.  In Restful web service, the payload can only be passed to the recipient through the POST method.

**Q #17) Enlist some of the HTTP methods with description.**

**Answer: Enlisted below is the list of HTTP methods with their descriptions:**

* **GET:** This is a read-only operation that fetches the list of users on the server.
* **PUT:** This operation is used for updating an old resource or for creating a new resource.
* **POST:** This operation is used for the creation of any new resource on the server.
* **DELETE:** As the name suggests, this operation is used for deleting any resource on the server.
* **OPTIONS:** This operation fetches the list of any supported options of resources that are available on the server.
* **PUT vs POST : An Example**
* Let’s say we are designing a network application. Let’s list down few URIs and their purpose to get better understanding when to use POST and when to use PUT operations.
* GET /device-management/devices : Get all devices
* **POST** /device-management/devices : Create a new device
* GET /device-management/devices/{id} : Get the device information identified by "id"
* **PUT** /device-management/devices/{id} : Update the device information identified by "id"
* DELETE /device-management/devices/{id} : Delete device by "id"

### 2. What are main differences between API and Web Service?

* All Web services are APIs but not all APIs are Web services.
* Web services might not contain all the specifications and cannot perform all the tasks that APIs would perform.
* A Web service uses only three styles of use: SOAP, REST and XML-RPC for communication whereas API may be exposed to in multiple ways.
* A Web service always needs a network to operate while APIs don’t need a network for operation.

well-known API’s: Google Maps API, Amazon Advertising API, Twitter API, YouTube API, etc.

### 7. What are the advantages of API Testing?

**Test for Core Functionality:**

***Time Effective***

***Early stage of testing***

### 3. What must be checked when performing API testing?

* Accuracy of data
* Schema validation
* HTTP status codes
* Data type, validations, order and completeness
* Authorization checks
* Implementation of response timeout
* Error codes in case API returns, and
* Non-functional testing like performance and security testing

### 24. What are API documentation templates that are commonly used?

There are several available API documentation templates help to make the entire process simple and straightforward

* Swagger
* Miredot
* Slate
* FlatDoc
* API blueprint
* RestDoc
* Web service API specification

### 34. What are the core components of an HTTP request?

An HTTP request contains five key elements:

1. An action showing **HTTP methods** like GET, PUT, POST, DELETE.
2. **Uniform Resource Identifier (URI),** which is the identifier for the resource on the server.
3. HTTP Version, which indicates HTTP version, for example-HTTP v1.1.
4. **Request Header,** which carries metadata (as key-value pairs) for the HTTP Request message. Metadata could be a client (or browser) type, format supported by the client, format of a message body format, cache settings, and so on.
5. **Request Body**, which indicates the message content or resource representation.