**End point:** Address where API is hosted on the Server.

HTTP methods which are commonly used to communicate with Rest API’s are

**GET, POST, PUT, and DELETE**

**GET**- The GET method is used to extract information from the given server using a given URI. While using GET request, it should only extract data and should have no other effect on the data. No Payload/Body required

**How to send input data in GET?**  
Ans: Using Query Parameters

**POST**- A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.

**How to send input data in POST?**  
Ans: Using Form Parameters /Body Payload

**PUT**- Replaces all current representations of the target resource with the uploaded content.

**DELETE**- Removes all current representations of the target resource given by a URI.

**Resources:  
Resources represent API/Collection which can be accessed from the Server**

Google.com/maps  
google.com/search  
google.com/images

**Path Parameters:**  
***Path parameters*** are variable parts of a URL path. They are typically used to point to a specific resource within a collection, such as a user identified by ID

<https://www.google.com/Images/1123343>  
<https://www.google.com/docs/1123343>  
<https://amazon.com/orders/112>

<https://www.google.com/search?q=newyork&oq=newyork&aqs=chrome..69i57j0l7.2501j0j7&sourceid=chrome&ie=UTF-8>

**Query Parameters:**  
Query Parameter is used to sort/filter the resources.

Query Parameters are identified with?””

https://amazon.com/orders?sort\_by=2/20/2020

**Headers/Cookies**:

Headers represent the meta-data associated with the API request and response. In layman terms, we were sending Additional details to API to process our request.  
Example : Authorization details

**End Point Request URL can be constructed as below**  
Base URL/resource/(Query/Path)Parameters

**API Contract:**

If we are asked to work on API testing, then we should ask for API contract which has information about the HTTP method, base URI, Query / path parameter, headers / cookies, authentication type, payload, response body etc., which will be provided by Development team.

**Dependencies to be added:**

Rest Assured, TestNG, Hamcrest from maven repository.

**How to build a basic test case:**

To test any API, first we need to declare the base URI using

Restassured.baseURL = “”.

Rest Assured is all about using three methods and to use them we need to manually write the import statement

“import **static** io.restassured.RestAssured.\*;” //used to solve given(), when(), then()

“import static org.hamcrest.Matchers.\*;” //to use equalsTo in assert that part in response validation.

given() – to pass all input details like headers, body

when() – to submit the api request with HTTP method and resource (will be passed as arg along with HTTP method)

then() - to assert the output

Also, we can add log().all() method next to given() and then() to give log information for the end user to understand the request and response body in readable format.

For assertion, there are plenty of things can be asserted such as status code, a key value pair from response body using body() and header content using header method()



Note: When you want to paste the request body as a string and for some reason if eclipse fails to format that properly, then check the following text box is enabled or not by navigating the path windows 🡪 preferences 🡪 editor.

A screenshot of a computer

Description automatically generated

**Parsing the response body**

There are sometime where we need to fetch the value from the response body and pass it as an argument to the subsequent request. To achieve this, we need to capture the response body as a string and using JsonPath class, we can parse it and get the desired key from it.

String response = *given*().log().all().header("Content-Type","application/json") .body(Payload.*addPlacePayload*()).when().post("/maps/api/place/add/json") .then().log().all().assertThat().statusCode(200).body("scope", *equalTo*("APP")) .header("Server","Apache/2.4.52 (Ubuntu)").extract().response().asString();

JsonPath js = new JsonPath(response);

js.getString();

**How to make use of mock response:**

When development team is still working on API but have the sample response body, then rather than waiting for the API contract, testing team can return the response body as a string from a method and use it in JsonPath js = new JsonPath(response)

To understand the nested json and array of objects in response body, make use of the following link

<https://jsoneditoronline.org/#left=local.fivisu&right=local.pibaya>

**How to send path param and query param as part of the request:**

Query param - can be sent using querypParam() along with given()

Path param – can be sent using pathParam() along with given()

given() .pathParam("**key**", issueId) .header("X-Atlassian-Token","no-check") .header("Authorization",”encode value of a api token”)

the same will be consumed in http method where we send the resource (because as per the end url, the path parameter is part of it, hence the best way is to append to resource) as below

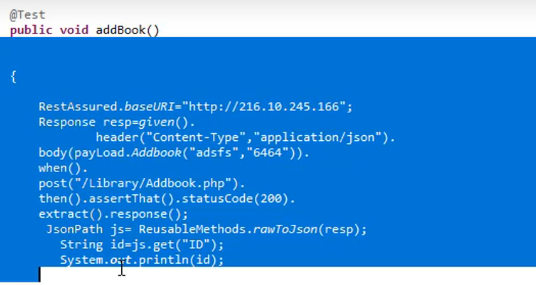
.post("rest/api/3/issue/{**key**}/attachments")

Note: the word key can be anything but it should be same in both place

**How to dynamically build json payload with external data inputs:**

1. **How to send parameters from payload:**

To send values dynamically (i..e, at run time) we can make the method which returns the value of the body to the test case takes 2 arguments and accordingly modify the payload to take 2 arguments based on the data type of test case and pass it in body.



A screenshot of a computer

Description automatically generated

If we run it once, it will pass but the next time it will throw an error stating that the book exists. To make it work continuously either we need to add a delete book test case next to it or keep changing the value of the parameter.

1. **Using data provider**

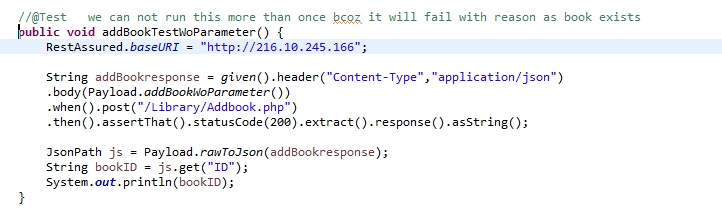
Create a method with data provider annotation and make the data

Provider to return 2d object array and each array consist of both isbn and aisle value. Give the knowledge of data provider to the test case using the name of the DP and make the test case method to take 2 arguments and pass it to the payload body method as shown below.



1. **Using static json payload**

Static json payload are something that the payload won’t change. For example, if you want to pull the number of accounts a particular bank branch holds then in this case, they payload can take the bank name, branch name and ifsc code, which will never change. In this scenario, we can make use of static payload.

To send static payload we can either directly pass the body in the test case directly or from another class by calling the static method 

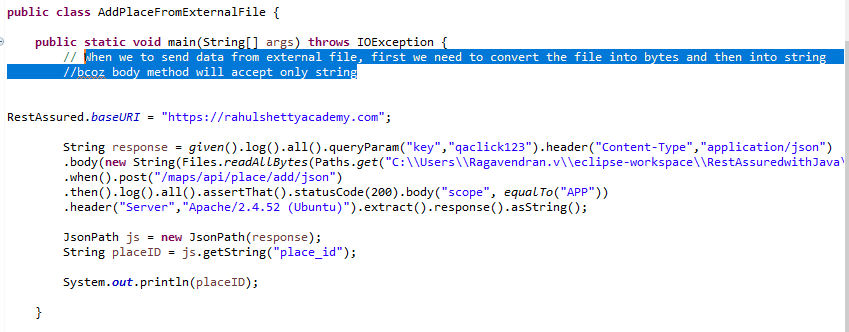
Another way is to use an external json file as shown below. When we want to send data from an external json file (we can use only when the payload is static), first we need to convert the file into bytes and then into string bcoz body method in given() will accept only string.

To convert into bytes

Files.readAllBytes(Paths.get(“path of the file”)

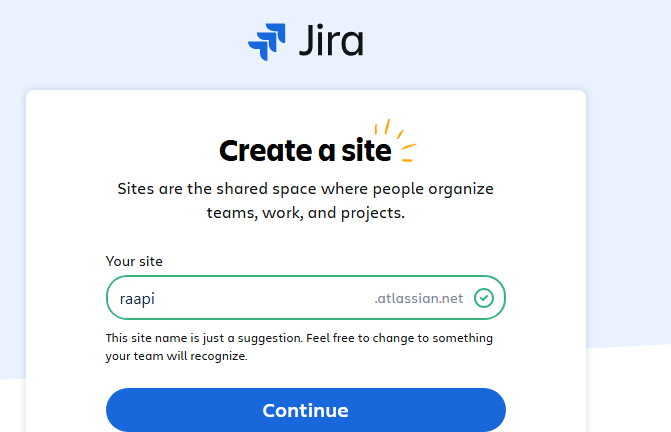
To convert into a string

New Sring(Files.readAllBytes(Paths.get(“path of the file”))



**To work with Jira API:**

[**https://www.atlassian.com/try/cloud/signup?bundle=jira-software&edition=free&application=wac--otp-jira&social-signup-login=true**](https://www.atlassian.com/try/cloud/signup?bundle=jira-software&edition=free&application=wac--otp-jira&social-signup-login=true)

Create an account using google and create a project by simply following the application flow. 

The cloud URL for JIRA where we are going to work using API is

<https://raapi.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog>

In order to work with JIRA like to raise a bug or to attach a screen shot to defect, the first step is to authenticate bcoz we are going to automate the stuff and the script should know to which user’s account, it needs to perform the operation.

Follow the link for cloud jira api documentation (**in this model, we are using basic auth**)

<https://developer.atlassian.com/cloud/jira/platform/rest/v2/intro/#authentication>

The user must create a api token first and followed by encrypting the token as mentioned in the documentation. Simple way is create the api token from the above link and store it in some place. Now in order to authenticate we need to create a encrypted key with the format as “email id used to create jira product: api token”.

<https://id.atlassian.com/manage-profile/security/api-tokens>

**Api Token is**

ATATT3xFfGF018XFwNnkyb1Y6tY4ksIkj6K7WT33G1xn62zQotHhVA475vKYBo78HmkPtALhurE\_vyMiblfVwZFDgMktBardigPcCHrE5SYNfrCPiI\_H6R0wD5V3xeE0V4cZGc4nU8Gr-JAerTO2MNePZXx8fFBs7pyJMQiIO93XRL2Ug8HhH3M=40A034A5

raghavdce@gmail.com:ATATT3xFfGF018XFwNnkyb1Y6tY4ksIkj6K7WT33G1xn62zQotHhVA475vKYBo78HmkPtALhurE\_vyMiblfVwZFDgMktBardigPcCHrE5SYNfrCPiI\_H6R0wD5V3xeE0V4cZGc4nU8Gr-JAerTO2MNePZXx8fFBs7pyJMQiIO93XRL2Ug8HhH3M=40A034A5

**Encoded base 64 API token is:**

**cmFnaGF2ZGNlQGdtYWlsLmNvbTpBVEFUVDN4RmZHRjAxOFhGd05ua3liMVk2dFk0a3NJa2o2SzdXVDMzRzF4bjYyelFvdEhoVkE0NzV2S1lCbzc4SG1rUHRBTGh1ckVfdnlNaWJsZlZ3WkZEZ01rdEJhcmRpZ1BjQ0hyRTVTWU5mckNQaUlfSDZSMHdENVYzeGVFMFY0Y1pHYzRuVThHci1KQWVyVE8yTU5lUFpYeDhmRkJzN3B5Sk1RaUlPOTNYUkwyVWc4SGhIM009NDBBMDM0QTU=**

As per the documentation, in order to create a bug, we need to authenticate as well as send the body for the post method in which one of the parameters is key. To get the value of the key, navigate Project 🡪 Click three horizontal dots on your project 🡪 click project settings 🡪 get the value of the key.

In this project, it is “SCRUM”

Body of create issue api is from

{

"fields": {

"project":

{

"key": "TEST"

},

"summary": "REST ye merry gentlemen.",

"description": "Creating of an issue using project keys and issue type names using the REST API",

"issuetype": {

"name": "Bug"

}

}

}

The end point will be domain name followed by resource name

End point - <https://raapi.atlassian.net/>

Resource - rest/api/2/issue

Base URL to create an issue in JIRA is - <https://raapi.atlassian.net/rest/api/2/issue>

Note: Header will be given in curl method of the documentation. It is used to add authentication as well as general headers

A computer screen shot of a computer code

Description automatically generated

So, to create an issue, the following headers will be added along with basic authentication

1. Authorization = Basic **cmFnaGF2ZGNlQGdtYWlsLmNvbTpBVEFUVDN4RmZHRjAxOFhGd05ua3liMVk2dFk0a3NJa2o2SzdXVDMzRzF4bjYyelFvdEhoVkE0NzV2S1lCbzc4SG1rUHRBTGh1ckVfdnlNaWJsZlZ3WkZEZ01rdEJhcmRpZ1BjQ0hyRTVTWU5mckNQaUlfSDZSMHdENVYzeGVFMFY0Y1pHYzRuVThHci1KQWVyVE8yTU5lUFpYeDhmRkJzN3B5Sk1RaUlPOTNYUkwyVWc4SGhIM009NDBBMDM0QTU=**
2. Content-Type = application/json

**Add attachment API**

<https://developer.atlassian.com/cloud/jira/platform/rest/v2/api-group-issue-attachments/#api-rest-api-2-issue-issueidorkey-attachments-post>

Base URL - [https://your-domain.atlassian.net/rest/api/2/issue/{issueIdOrKey}/attachments](https://your-domain.atlassian.net/rest/api/2/issue/%7bissueIdOrKey%7d/attachments)

Note: User can get the issue id from create issue response.

A close-up of a computer screen

Description automatically generated

When we want to send any file as attachment in rest api, then we need to make use of “form-data” parameter under body as shown below in postman.

A screenshot of a computer

Description automatically generated

The form-data will take key value pair, where it takes file as a key and the value as path of the file and the key type is “file” from the dropdown.

Interview question – When you want to send something securely, then will you use query parameter or form parameter. The answer should be form parameter using form-data from body and not from query parameter because it will go with URL which is not safe.

**With Rest Assured to do an attachment – use multipart()**

To make an attachment to the existing issue, we need a screen shot image either in pdf or in jpg or in png format and a issue id.

To attach a file in the rest assured api request, we need to make use of multipart() method() as shown below which takes 2 parameters i.e. 1. File and 2. New File(Path)

A computer code with blue text

Description automatically generated with medium confidence

**OAuth in Rest Assured:**

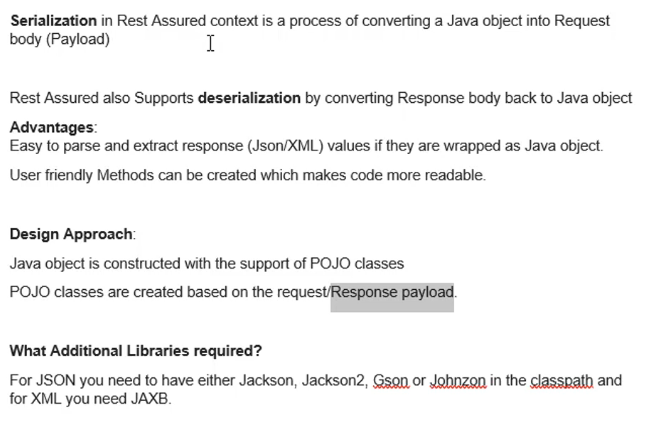
Let us take a banking application as soon as we enter credentials and hit enter, the application shows us the home screen along with information like balance, credit card, loan, profile information etc.

We assume it happening with just one authentication i.e. at the login page but there is a second level of authentication happening between API. This is where we are using OAuth authentication which enables authentication between servers / API.

Generally, there are three types of grant level is used to make this concept work, and they are **1. Client credentials (using client id and client secret), 2. Password (username and password) and 3. Authorization code.** We need to check with Development team about what kind of authentication are they using and if they say OAuth then we need to ask about the grant type and the required details based on the type.

Now to make the communication happening between the servers, the API should get an authentication token which will be provided by Authorization server sits between the API. With the help of this token, we can interact with other API such as bank balance, credit card information, personal loan etc.,

**Serialization and Deserialization of Request and Response using POJO classes:**



In general, we send the body directly as part of the given() or from external method or from external file. But using serialization concept, we can simply pass the java object reference into the body, and rest assured will understand the pojo class object and convert them into json.

**How to achieve serialization using POJO class:**

Let us say we need to pass the following body

{

“message”:”hello”,

“greet”:”hi”

}

Create a class and create getters and setters for all the key available in your json body in this case message and greet are the keys.

Public class serialization{

Private string message;

Private string greet;

Public String getmessage(){

Return message;}

Public void setMessage(String message){

This.message = message;}

//similarly for the other variable greet

//now create the object for this class using

Serializtion obj = new Serializtion();

Obj.setMessage(“hello”);

Obj.setGreet(“hi”);

//to send it as a json body in rest assured request

Given().body(obj);

**Deserialization:**

It is opposite of serialization where we convert the response into a java object and using getters methods, we will get the desired value. Here also we need to create a pojo class similar to the serialization but with the variables of the response received. Then create setter and getters for the same. Using java object reference and with getters method we can fetch the value of the desired key in the response and by this we can eliminate Jsonpath class requirement.

To achieve serialization and deserialization in rest assured, we need to add couple of dependencies such as Jackson, jackson2, Gson for JSON body and alternatively we can simply use Jackson-databind as shown below in pom.xml

A screenshot of a computer

Description automatically generated

**How to create POJO classes for the JSON body:**

Create separate class with all keys in the JSON body as a separate private variable and create getters and setters’ method for all if it is straight forward string. If it is of type nested JSON, (please refer the JSON body below to understand the structure) where courses are a nested JSON.

So, in this case we need to create a separate class for the child JSON and make the courses variable in first class data type as courses (name of the sub-JSON class).

In turn, if the sub-JSON is of type array then again create separate classes for all (in this case webautomation, api and mobile) and declare the keys (courses title and price) inside it as separate private variable. Since now it is a array, then we need to make the data type of courses class as List<WebAutomation> webautomation, List<API> api, List<Mobile> mobile etc.,

A screenshot of a computer

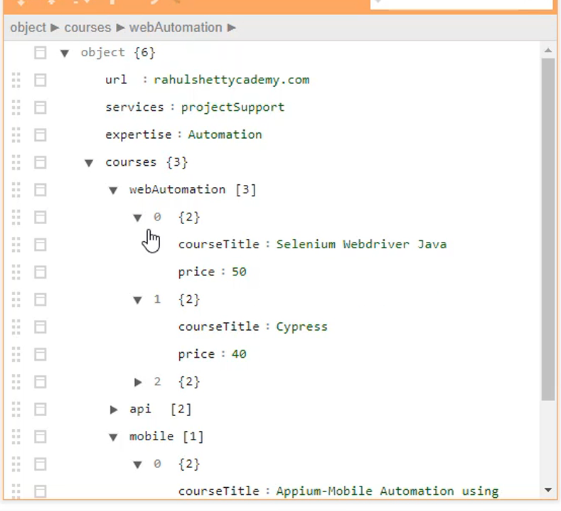
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



**How to access the response and get the key and it values using Deserialization:**

So far we have written in the response to get it as string and then pass it to JsonPath class and get the value using getString or getInt method. But after we implemented the POJO class, it will be easy that we can save the response as parent class(GetCourse) and store that into the parent object as shown below.

Note the return type is now Getcourse object and not string.

A close-up of a computer screen

Description automatically generated

**How to get the content of a particular key when JSON is nested and static:**

Let us say we want to get the value of the course details under API whose value is “SoapUI Webservices testing”, then to iterate through the path we can write

Gc.getCourses().getAPI().get(1).getCourseTitle();

Note: since getAPI return type is list, we can call the get method on it.

**How to get the content of a particular key when JSON is nested and dynamic:**

In this case, the position of the particular key keeps varying based on new keys are added or removed from the existing list. In this case, lets get the list of course and store into a list variable and the iterate through it using for loop and with if condition we can get the desired value. Let us the user wanted to get the price of “Rest Assured Automation using Java”.

List<API> list = gc.getCourses().getAPI();

For(int i=0;i<list.size();i++){

If(list.get(i).getCourseTitle().equalsIgnorecase(“”)){

System.out.println(List.get(i).getprice()):

} A screen shot of a computer code

Description automatically generated

**Serialization Example:**

All this time we used to send the JSON body as part of the request is by directly copy pasting the JSON boy but with serialization we can create the java object to construct the JSON body and pass the object reference in the request body.

First we need to create POJO classes for all the key available in the request body. If it is of type

1. String 🡪 private string variable name;
2. Array 🡪 private List<String> variable name;
3. Sub JSON 🡪 private class name as return type variable name;

For string straight away create getter and setter and for array set the type as List<String/In> based on the variable type and then create getters and setters. If a key as sub JSOn, then create a new class and have the variable inside the sub json and create getters and setters. Then change the data type of parent key as child class name.

Then using setters method set values. for string it is easy whereas for array, we first need to create an array list and add all the variable name and then pass the list reference to the setters method. Then for sub JSON, we need create object of the sub json and set variable for the sub json setters method and then pass the sub json object reference in the parent json setters method. Please see the screen shot below to understand better.

Note: In this scenario, for array it is not in key value pair but just a string. Assume if we have key value pair, then we must have created one more class and make the list type as List<newClassCreate>. Also, to add using setters method, if we have more than one array, then first create multiple objects for that class and keep adding for each array. Then create a new array list and add all the objects to it.

A computer screen shot of a code

Description automatically generated

A screenshot of a computer code

Description automatically generated

**Request and Response Spec Builder:**

In general, when we send request as well as when we get response, there are some common steps that is involved like with request we will send header, query param, path param, authorization etc. Similarly in response we check the content type and status code.

Instead of repeating these pieces of code again and again, we can write them in one place and keep using it for the subsequent steps. To achieve this, we can use either request or response spec builder.

Req = new RequestSpecBuilder().setContentType(“application/json”).setBaseUr(“”).addQueryParam(“key”,”qaclick123”).build();

Similarly for response, combine status code

To use it in the request we can simply write it as

Given().spec(req).when().post(“”).then().

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated



**E-Commerce portal Automation using API with token authentication:**

**URL -** [**https://rahulshettyacademy.com/client/auth/login**](https://rahulshettyacademy.com/client/auth/login)

**UN -** [**raghavdce@gmail.com**](mailto:raghavdce@gmail.com) **and PWD - Rahulshetty@123**

**Login API:**

**End point –** <https://rahulshettyacademy.com/api/ecom/auth/login>

**Payload** - {"userEmail":"raghavdce@gmail.com","userPassword":"Rahulshetty@123"}

**Response –**

**{**

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NThhN2Y4MTlmZDk5Yzg1ZThmM2YwMmEiLCJ1c2VyRW1haWwiOiJyYWdoYXZkY2VAZ21haWwuY29tIiwidXNlck1vYmlsZSI6OTk2MjM4NTg1NCwidXNlclJvbGUiOiJjdXN0b21lciIsImlhdCI6MTczNzI5MzExNSwiZXhwIjoxNzY4ODUwNzE1fQ.RW7O93eX4aqOgwqrZEguSUx1s-7E\_uUaQXBtMDscF9M",

"userId": "658a7f819fd99c85e8f3f02a",

"message": "Login Successfully"

**}**



**To add a product to the web page:**

A screenshot of a computer program

Description automatically generated

**How to solve SSL Certificate trust issue:**

Use “relaxedHTTPSValidation()” method next to given(). This will handle the SSL certificate issue.