# API Testing Basic theory

* Http Basics
* XML & JASON
* Soap and Rest API
* Why API Testing
* API Testing Tool
* API stands for Application Programming Interface
* API has three major parts: the request, the program and the response.

Different between API and Web services

* Web services is an api that goes throw the internet.
* Api is connected to the internet is api called web services.
* If this is an offline API that is not connected on the internet this is called only an API.
* So after all web services are API because a web services is a special type of an API.
* But not all APIs are web services because some of the api are not on the internet we can’t call them web services.

When we are dealing with online **APIs or Web services** we use two types of formats to format data, **XML and JSON.**

**And** The Protocol or the way that use to transfer XML or JSON data is SOAP or Rest APIs

Different between XML and JSON

* XML for **extensible Markup Language**

**What is XML?**

* Extensible Mark-up Language
* A format we use to send the APIs
* Created by W3C(The same organization created HTML)
* In XML the terms in the tags don’t mean anything
* In HTML the terms mean something
* HTML is not Extensible (It can’t extend it to mean something else)
* Brower can understand XML
* For send some XML data from HTTP Request like
* HTTP header Line: Content-Type: application/xml
* HTTP Body: XML

**<Course>**

**<Category>**Software Development**</Category>**

**<Section>**Software Testing**</Section>**

**<Target audience>**

**<aud>**students**</aud>**

**<aud>**Software Engineer**</aud>**

**<aud>**Junior Testers **</aud>**

**</Target audience>**

**<Section>**Software Testing**</Section>**

**<Target audience>**

**<aud>**students**</aud>**

**<aud>**Software Engineer**</aud>**

**<aud>**Junior Testers **</aud>**

**</Target audience>**

**</course>**

**What is JSON?**

* JavaScript Object Notation
* It’s the part of JavaScript that holds data
* JSON code is smaller than xml
* For send a JSON code with request write like
* HTTP Header Line: Content-Type: application/json
* HTTP Body: json

{“Course”:

{“Category”: “ Software Developmet”,

“Section”: “Software Testing”,

“Target Audience”: [“Student”, ”Software Engineer”, ”Junior Tester”]

}

}

|  |  |  |
| --- | --- | --- |
|  | XML | JSON |
| Powerful | **YES** | **NO** |
| Simple | **NO** | **YES** |
| Developed | **1997** | **2001** |
| Popularity | **Down** | **Up** |

Different type of APIs which two major type of APIs

* SOAP
* REST

So, What is SOAP?

* Simple Object Access Protocol but its old
* It uses for WSDL (Web service Description Language) but also this is not description language for web services but that time it was only type of web service Description Language.
* For using **HTTP** in this case **start line** will be for
* POST WSDL HTTP Version
* **Header Line** will be content type will be
* Content type: text/xml
* Blank Line
* Body
* Will written in XML envelope formed using WSDL

So the body is SOAP APIs is created by XML using the standard which is WSDL,

What is REST APIs?

* REST stand for Representational State Transfer
* It is simpler than SOAP APIs, it’s easier.
* Rest is Representational, which means the actual record is not sent, a representation of the record is sent
* Rest is Stateless it means that is does not matter the status of the system at this moment. Just send request and get the response even if the server has some problems ( but in the soap APIs, If the program or server has any problem, it will breakdown and this will cause a major issue. )
* Rest APIs waits until it work (stateless)
* Rest uses JSON

In Rest APIs we have many types of requests

|  |  |
| --- | --- |
|  | API order |
| Post | Create an Order (JSON) |
| PUT | Erase the older order and create another one |
| Patch | Keep the old order and add another one |
| Get | Receive the order details |
| Delete | Remove the order |

Most Important ones are **post, get, and delete.**

So, for **POST** I need to got to a website and I want to create an order with a JSON body.

Byusing **PUT** create an order and want to edit also in this case will use **put. (**delete old data and add other data**)**

Like for **PATCH** I have folder for example any name, I will add another folder next to it.

For **GET** receive my order details or anything that is on the server.

And Of course, **DELETE** will remove the order or the action that I already did.

API tool like **Postman** example, most of the work is done using rest APIs

Most important principles in software testing is **early testing**

* This will save time and money
* Push the testing toward the lower test levels is good.
* We can say we have 3 test levels,

1. Unit/component testing
2. APIs Testing
3. GUI/UI testing

**GUI/UI** testing is called sometime **system testing**.

**API** testing, it’s considered as a type of **integration testing.**

**Developer** perform unit/component testing weather manual or automated in order to find defects as early as possible.

# Postman

# Section Content:

* What is Postman.
* Installation process
* Request Builder
* Create Request
* Save Request
* Writing Tests
* Postman is a tool that use to test own Rest APIs. Or external APIs (FB, Google, etc).
* Postman is used by 6 million developers and more than 200,000 companies to access 130 million APIs every month.
* Postman is a browser.
* Installation will be done as normal process.
* Request Builder
* Request builder is main section of postman.
* **GET** as example
* httpbin.org then if we click on send, we will find a http response and where we can see in **Body** section **Pretty, Raw, Preview** with **Status, time, size**
* And If we add **GET** with https://httpbin.org/get then we will found the **body** is different with comes appears with JSON.
* And if we want to send some parameter like **KEY** as P1 and **VALUE** as testing And **KEY** as P2 and **VALUE** as request
* **If** edit them in **URL** then it will automatically edit in **Query Params** and Lets **send** then it will added some section in “args” in JSON Body.
* After creating request, we can **Save** it and give a **Request name** Like “**Get API Request**” and It must be inside a collection if have no any collection then need create a collection like “**API Testing Collection**”

**Also,** we can see the history by left side of **Request builder.**

* **POST** as example
* Change it to **POST** URL with httpbin.org then go to **Body** and click on **raw** then there have to write the body of request which has been sent as a **post** most of the time is **json. Like**
* {
* "name":"Taief",
* "Course":"SQA"
* }
* If we **send** the data the request will be failed because the page has been designed only show some data
* So that’s why have to go with another page with **post** httpbin.org/post then click on send we will see the data insite response of my request.
* Okay Lets now create some test for these **two requests** Which is **GET** and **POST**
* First go for **GET** request and we will click on test So If we go for **Right** of **request builder** which is called **SNIPPETS PANNEL** Therehave some pre written test that I can use inside my program.

As example that have **Status code: Code is 200** if we click on it there will show a **function** in **test panel** like

pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);

});

As a result, she will check that the **response** is **equal 200** or not.

If it is then we will see the test result is **pass.**

And If we change it to **200** to **400** it will show test result is failed. Will appear an **AssertationError**: expected response to have status code 400 but got 200.

Also, can say needs to make sure the **response time is less then 200 millisecounds.**

Like

pm.test("Response time is less than 200ms", function () {

    pm.expect(pm.response.responseTime).to.be.below(200);

});

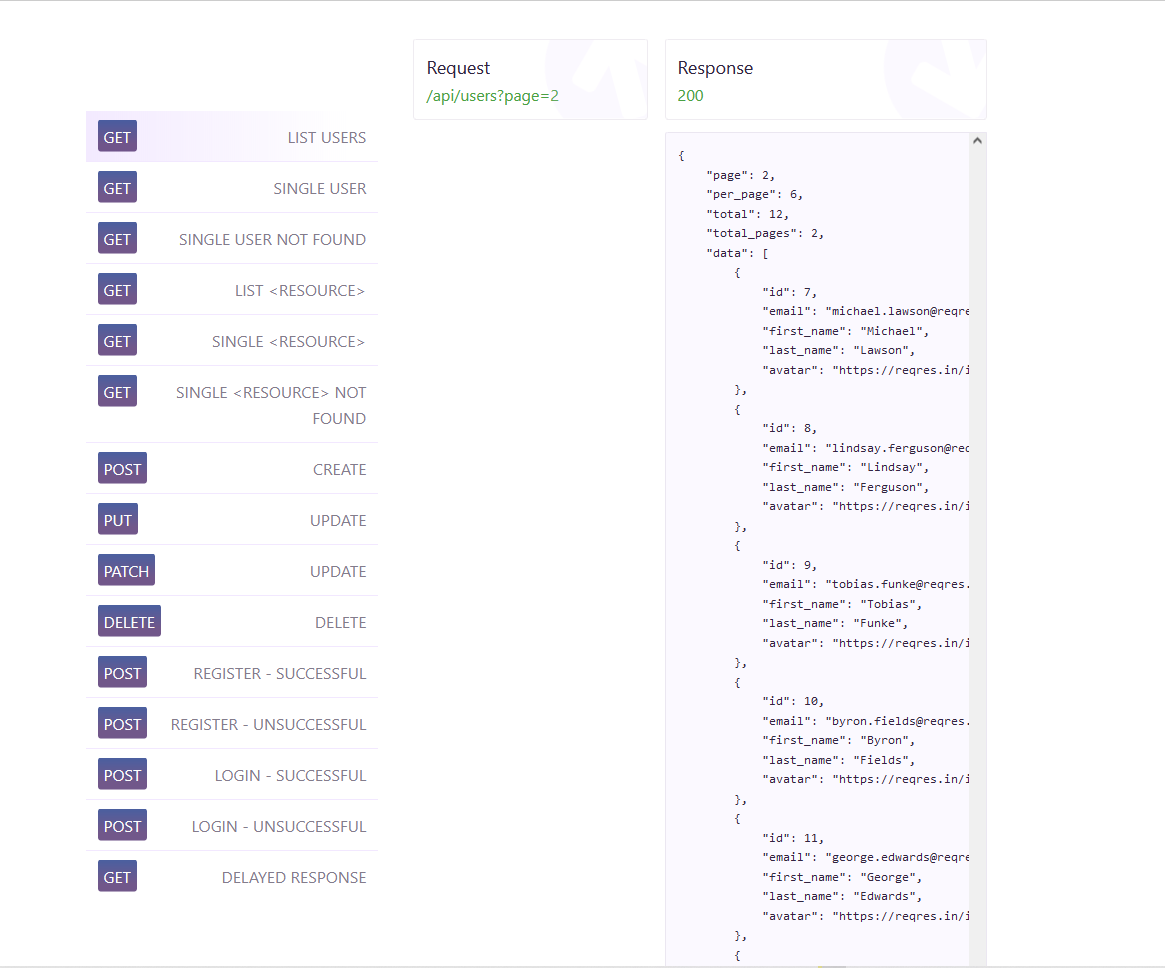
If it is pass then will will show okay otherwise it will show an   
**AssertionError:** expected 622 to be below 200 And **Save**

* And if we go for **POST**

For **post** we will write the same test.  
The **pass/fail** result will show in test result as like as before.

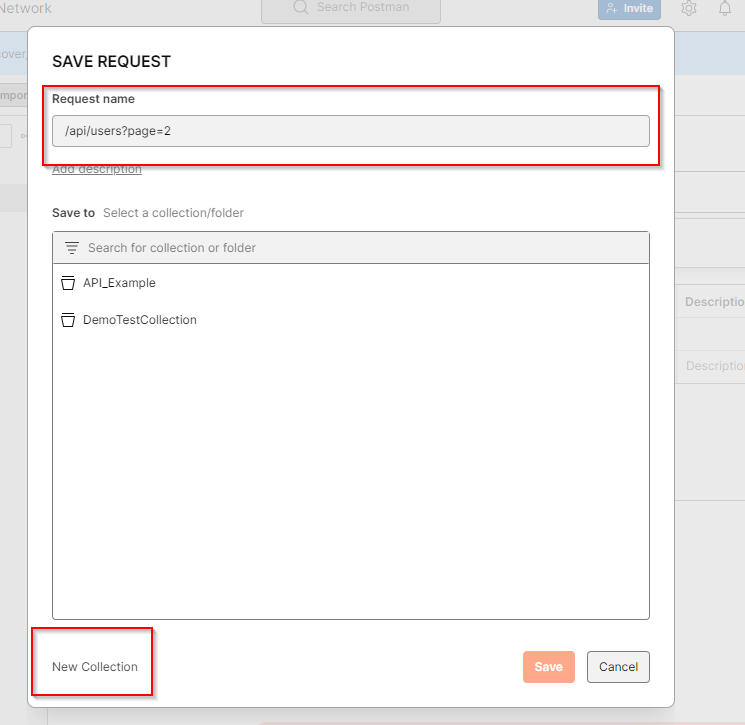
* **Collection Runner:**
* If a scenario appear that need to test many requests sequentially not manually that’s why it have something that’s called collection runner.
* So, we must click on the **three dot “**it’s had every side each **collection**” menu and click on **Run collection**
* And there must set Choose **how run your collection** if it is run manually then check **run manually** and **iterations** (How many times it will occur) Delay (like **1000ms = 1Sec**) and **check the box** as need.
* Then have to click the **run collection** and it will run as the as commanded.
* **Doing Test which has been previously run with a simple project website called** <https://reqres.in>
* Navigate This to this website there are some free APIs that can user for testing.
* So, let’s try So that we can know how we learn on them.
* So, In real projects we get something from developer which is called API Documentation.
* So API documentation is document like this <https://reqres.in> pages that describe The APIs and what is the expected behavior of the APIs.
* So,as a tester here have to play the role is to make sure that the real behavior, or the actual behavior is the same as the expected behavior that is stated inside the document.

Image 1.0



* we can see on image 1.0 if we go for **GET** there is a request for get which says the **LIST USERS** AND called **Request** is the api end point and the **Response 200** so there together is the **expected response** not the actual response. And, the left side body is **expected body.** That should get when you send this request.
* So there is 15 or 12 requests so we are going to test them, run them manually, add tests to them.
* After that then begin to run them automatically using the collection runner.
* So lets begin by the first request So we will open **Postman** first.
* After we go for first request
* lets begin by writing it [**/api/users?page=2**](https://reqres.in/api/users?page=2) **but if we just go and send with this url it will say Could not get any response or could not send request. So why? Because this is not URL so we have the host or the base URL. Then we have to add this path. So here it means that you should ad the homepage or the URL of the home page before any request. So we should add this before the request https://reqres.in**[**/api/users?page=2**](https://reqres.in/api/users?page=2) **so here it means that you should add the homepage or the URL of the home page before any rerquest.**
* **So now we are going to add beside GET request like this https://reqres.in**[**/api/users?page=2**](https://reqres.in/api/users?page=2)
* **So in any API request that we receive from the developer for example, we should assume that we will need to add the base url. Like facebook.com or google.com the website that we are developing.**
* **So now if we click send then we will check the status, time and the Actual body.**
* **So in Actual body we will the “page” : 2, “per\_page”: 6, “total”: 12, “total\_pages”, data from “id”: 7 so all their behavior actually as the document above called expected behavior or not.**
* **So if everything is alright we can click ctrl + s to save it. We will see a pop-up like this.**

**Image 2.0**

* 
* So removing the url from Request name we can give them a name like **List Users** or any name we can prefer.
* Then we will create a new collection and call it **APIs\_example or** any preferred name. then Save it to **APIs\_example.**
* Then we can see right site of the **Request builder** in collection called **APIs\_example**.
* So if we click on it contains a request which was we saved.
* So now if we run **collection runner** by clicking three dot (…) like we said before we will not see any result. If we run it with **1 iteration** then we will see 0 passed and 0 failed. So Why?
* Because we did not add any tests to this request.
* So now its time to add test this request.
* So we will click on **GET** which one we saved already.
* Then we have to go Scripts and post response and There we can see the right site call **SNIPPETS**
* So there we will **see status: code is 200**. So we click on it and **save.**
* Then if we click on send we will see In **Test Result PASS** Status code is 200
* So we can test from for the response time so we will click again form **SINNPET** called **Response time is less than 200ms** or we can say less than **1000ms = 1s.**
* and we can change the message in double quotes instead of “Status code is 200” we can say “users are listed successfully” so here we got two test cases lests click on **send** again we will see Test result **pass** user are listed successfully and **Pass** Response time is less than 1000ms .

**Performance Testing**

* **Time Behavior**
* **Resource Utilization**
* **Capacity**

**And Types of Performance Testing**

* **Load Testing**
* **Stress Testing**
* **Scalability Testing**
* **Spike Testing**
* **Endurance Testing**
* **Concurrency Testing**
* **Capacity Testing**