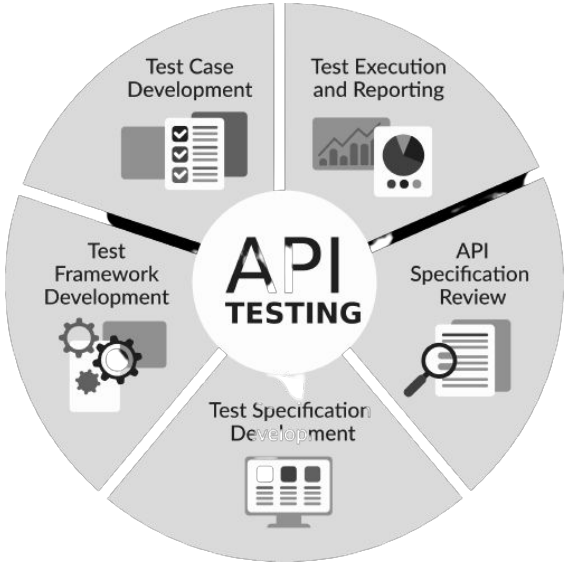
API Testing Mastery



Pramod Dutta
Lead SDET.

Web Fundamentals for API Testers

Agenda of Session

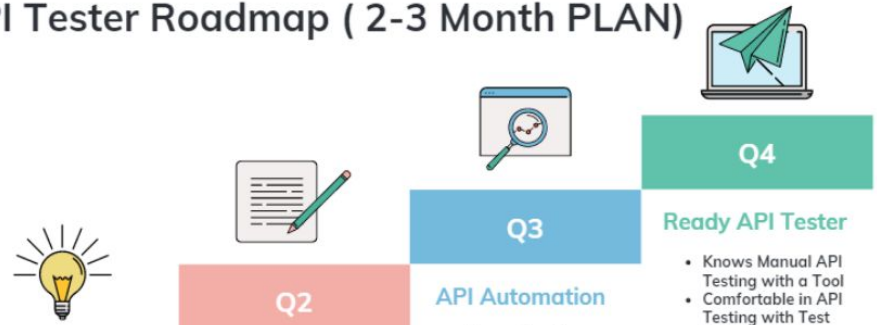
- HTTP Basics, URL Basics
- Cookie, Headers
- JSON Basics & XML Basics
- Install Tools for API Testing
- Chrome Dev Tools, Import Requests
- HTML Forms for API Testing

API Tester ROADMAP

https://apitesting.co/blueprint

Made by Pramod(TheTestingAcademy

API Tester Roadmap (2-3 Month PLAN)



Q1

Web Basics+ Mindset

Web Fundamentals
API / Web Services
API Testing What, why How?
API Testing Tools.
Action Plan for 2-3 Months

Q2

Manual API Testing

- Types of API Testing.
- SOAP Vs Rest, GraphQL
- POSTMAN
- Advance Postman with DDT, JSON Schema validations
- LIVE Projects
- Test Design, Bug reporting, Debug APIs Web dev Tools
- Advance Web Auths extra

Q3

API Automation

- Learn Core Java
- Learn about Maven, TestNG
- Rest Assured
- Framework Work
- Manual Live Projects.
- CI/CD Tools like Jenkins, Git, Docker

Q4

Ready API Tester

- Knows Manual API Testing with a Tool
- Comfortable in API Testing with Test Design, Planning and Releasing.
- Know how to automate API with framework.
- Have an idea about common 100+ API Testing Interview QnA.
- Added projects to Github & Resume.

https://apitesting.co/blueprint

API Tester ROADMAP

https://miro.com/app/board/uXjVOttFRjM=/?share_link_id=284599875783

Know Your Instructor -Pramod Dutta

- I have nearly 9+ year of Experience in the Software Testing.Worked on different domain Mobile App, Web API, Website , Performance testing etc.
- Taught 10K+ Students
- 8600 Students in FB, 55k Youtube Subs.
- Helping People to Become Better Automation Tester.
- contact@thetestingacademy.com

Wingify BrowserStack accenture

Before We Start.....
Join Private Telegram

https://sdet.live/apibatch

Do you know What is HTTP?

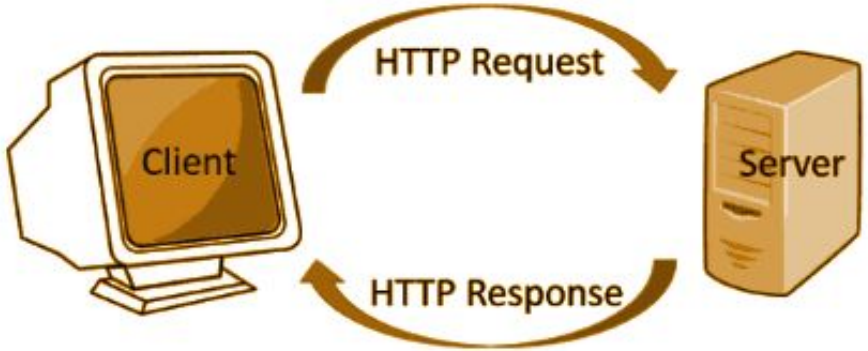
Yes

No

In Comments

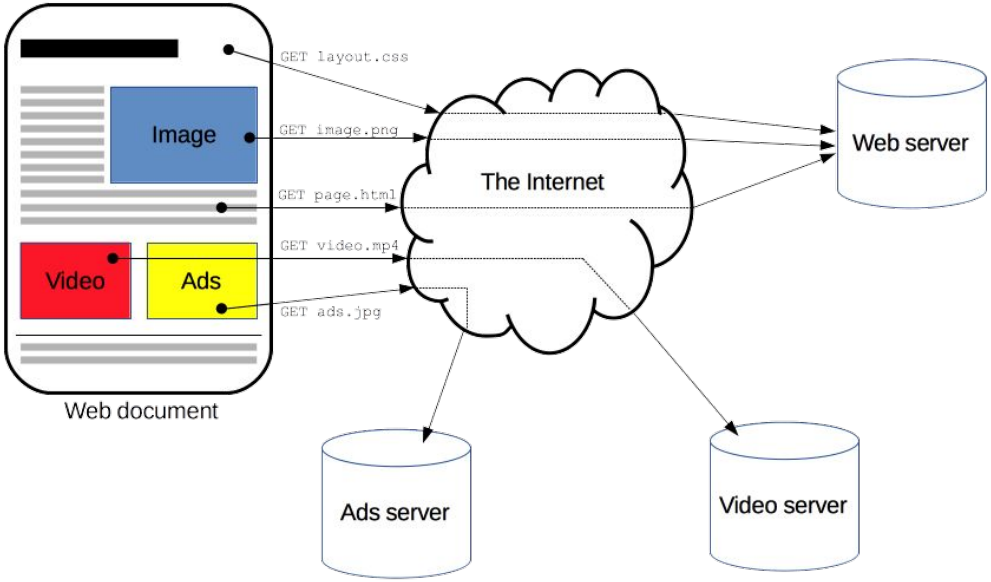
Overview of HTTP

- HTTP is a protocol for fetching resources such as HTML documents.
- It is the foundation of any data exchange on the Web and it is a client-server protocol.
- HTTP is stateless: there is no link between two requests being successively carried out on the same connection
-

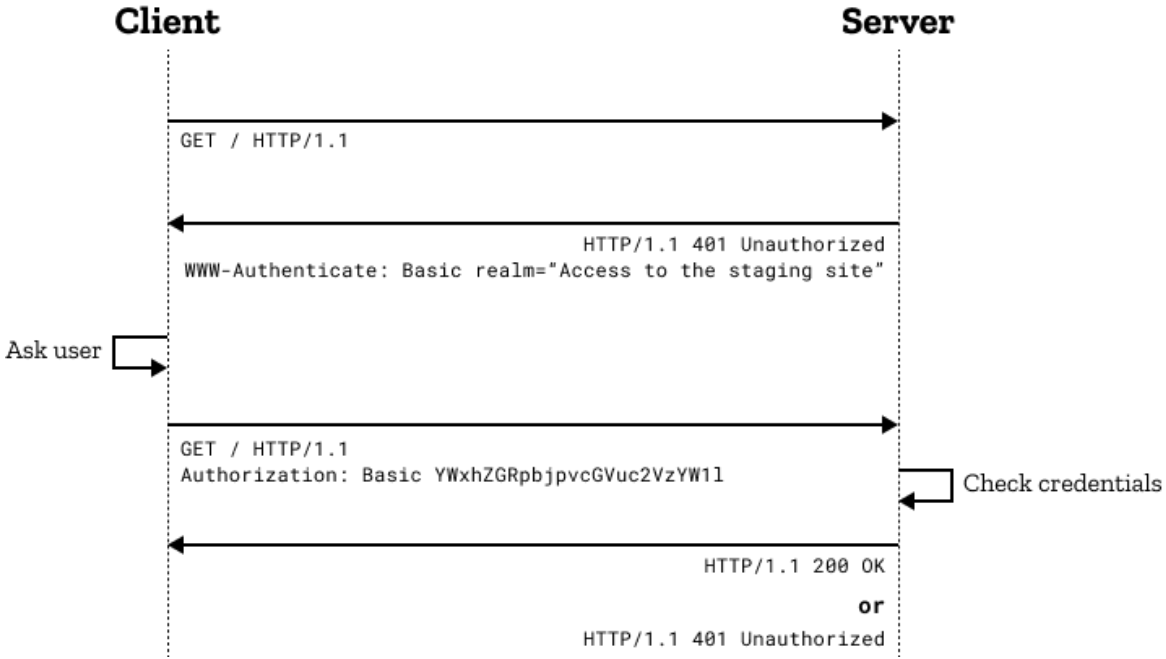


https://developer.mozilla.org/en-US/docs/Web/HTTP

Overview of HTTP



HTTP Authentication



HTTP Cookies

An HTTP cookie (web cookie, browser cookie) is a small piece of data that a server sends to a user's web browser.
The browser may store the cookie and send it back to the same server with later requests

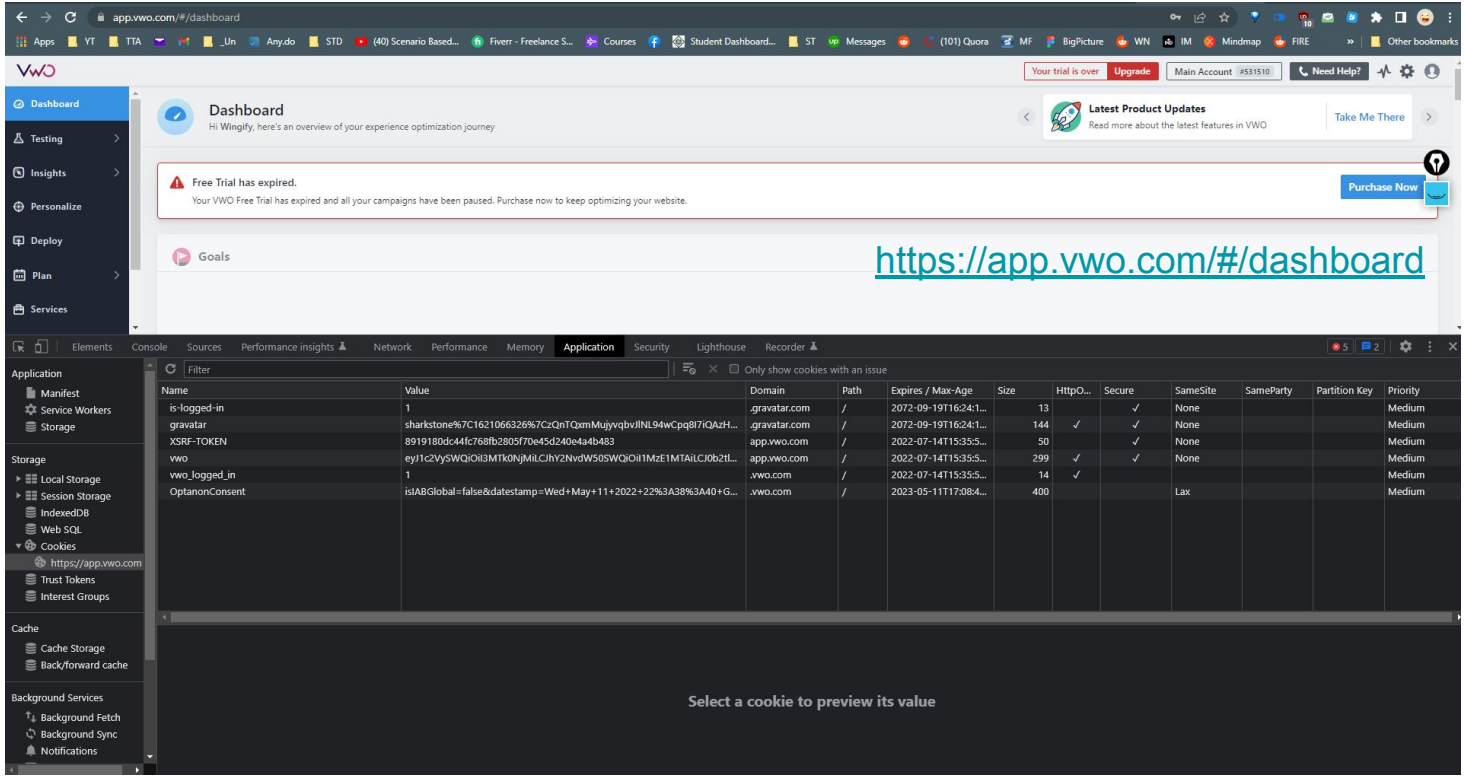
Cookies are mainly used for three purposes:

Session management
Logins, shopping carts, game scores, or anything else the server should remember

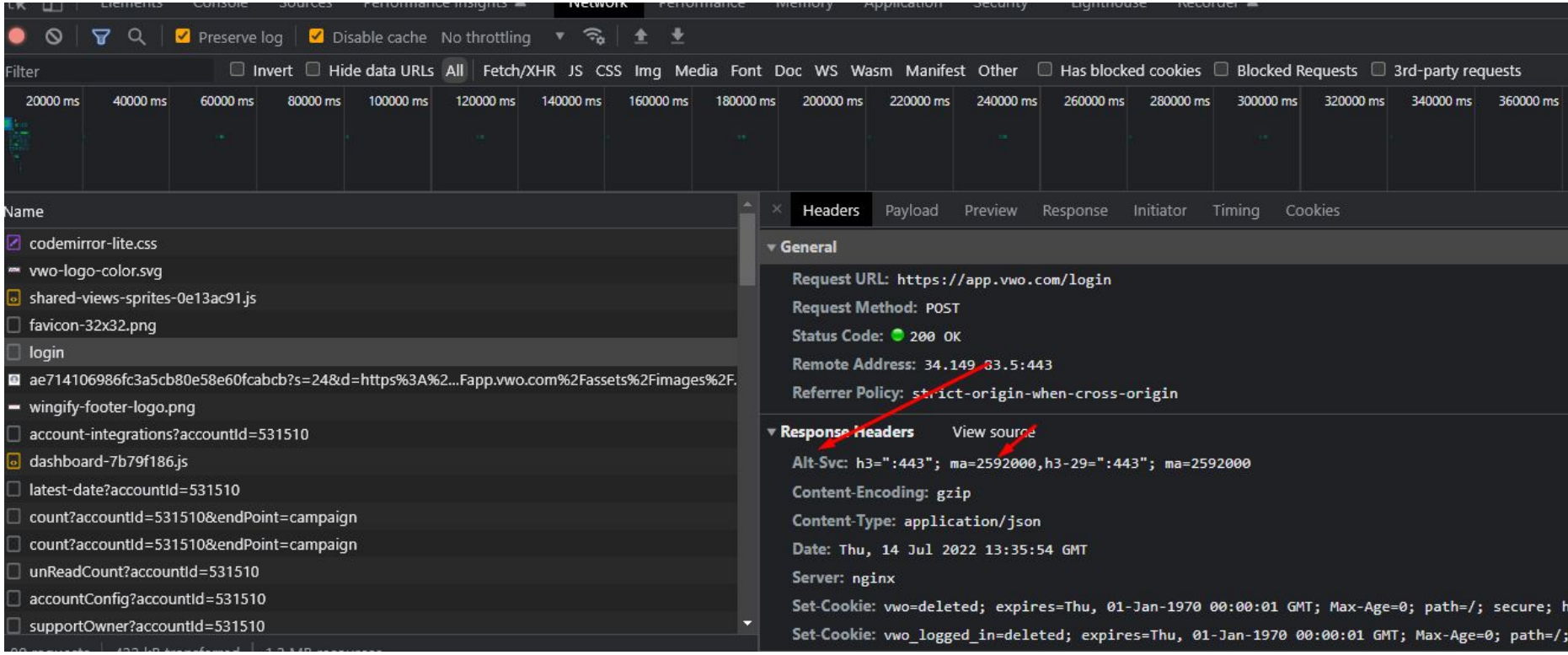
Personalization
User preferences, themes, and other settings

Tracking
Recording and analyzing user behavior

HTTP Cookies



Headers


Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Font
Doc
WS
Wasm
Manifest
Other
Has blocked cookies
Blocked Requests
3rd-party requests
20000 ms
40000 ms
60000 ms
80000 ms
100000 ms
120000 ms
140000 ms
160000 ms
180000 ms
200000 ms
220000 ms
240000 ms
260000 ms
280000 ms
300000 ms
320000 ms
340000 ms
360000 ms
Name
codemirror-lite.css
vwo-logo-color.svg
shared-views-sprites-0e13ac91.js
favicon-32x32.png
login
ae714106986fc3a5cb80e58e60fcabcb?s=24&d=https%3A%2...Fapp.vwo.com%2Fassets%2Fimages%2F.
wingify-footer-logo.png
account-integrations?accountId=531510
dashboard-7b79f186.js
latest-date?accountId=531510
count?accountId=531510&endPoint=campaign
count?accountId=531510&endPoint=campaign
unReadCount?accountId=531510
accountConfig?accountId=531510
supportOwner?accountId=531510
Headers
Payload
Preview
Response
Initiator
Timing
Cookies
General
Request URL: https://app.vwo.com/login
Request Method: POST
Status Code: 200 OK
Remote Address: 34.149.83.5:443
Referrer Policy: strict-origin-when-cross-origin
Response Headers
View source
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
Content-Encoding: gzip
Content-Type: application/json
Date: Thu, 14 Jul 2022 13:35:54 GMT
Server: nginx
Set-Cookie: vwo=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/; secure;
Set-Cookie: vwo_logged_in=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/;

Headers

HTTP headers let the client and the server pass additional information with an HTTP request or response.

- Request headers contain more information about the resource to be fetched, or about the client requesting the resource.
- Response headers hold additional information about the response, like its location or about the server providing it.
- Representation headers contain information about the body of the resource, like its MIME type, or encoding/compression applied.
- Payload headers contain representation-independent information about payload data, including content length and the encoding used for transport.

HTTP header indicates which content types, expressed as MIME types, the client is able to understand.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

Authentication

Caching

User agent client hints

Conditionals

Headers

The Content-Type representation header is used to indicate the original media type of the resource (prior to any content encoding applied for sending).

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type

Tools for Cookie, Headers View

https://chrome.google.com/webstore/detail/editthiscookie/fngmhnnpilhplaeedifhccceomclgfbg?hl=en

URL Basics

URL BASICS

Uniform Resource Locators

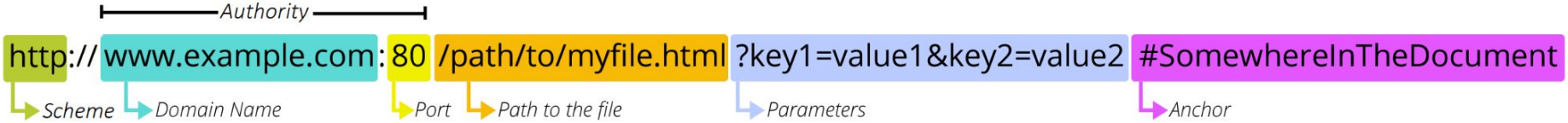
It is the mechanism used by browsers to retrieve any published resource on the web.

A URL is nothing more than the address of a given unique resource on the Web

Such resources can be an HTML page, a CSS document, an image, etc

```
https://developer.mozilla.org
https://developer.mozilla.org/en-US/docs/Learn/
https://developer.mozilla.org/en-US/search?q=URL
```

URL BASICS

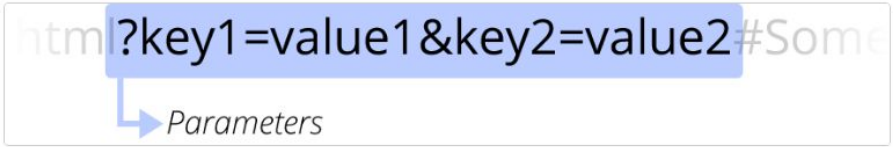

Authority
http://
www.example.com
:
80
/path/to/myfile.html
?key1=value1&key2=value2
#SomewhereInTheDocument
Scheme
Domain Name
Port
Path to the file
Parameters
Anchor

URL BASICS

Path to resource

`/path/to/myfile.html` is the path to the resource on the Web server. In the early days of the Web, a path like this represented a physical file location on the Web server. Nowadays, it is mostly an abstraction handled by Web servers without any physical reality.

Parameters



Quiz

https://abc.com/xyz/?q=1

Which one is path param
Which one is query param

Absolute URLs?

What are Absolute URLs?

An absolute URL contains the entire address from the protocol (HTTPS) to the domain name (www.example.com) and includes the location within your website in your folder system (/foldernameA or /foldernameB) names within the URL.

Basically, it's the full URL of the page that you link to.

An example of an absolute URL is:

<a href = **http://www.example.com/xyz.html**>

What are Relative URLs?

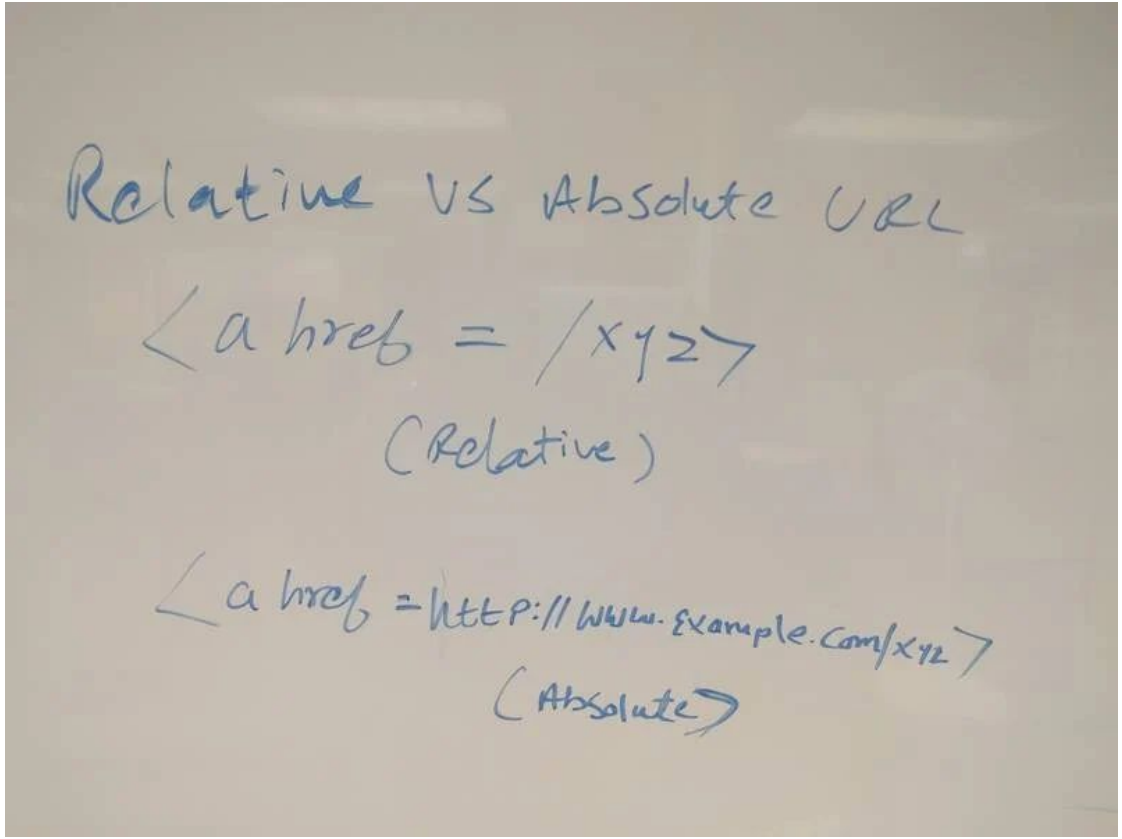
What are Relative URLs?

The relative URL, on the other hand, does not use the full web address and only contains the location following the domain. It assumes that the link you add is on the same site and is part of the same root domain.

The relative path starts with the forward slash and leads the browser to stay within the current site.

An example of a relative URL is:

<a href = "**/xyz.html**">


Relative VS Absolute URL
<a href = /xyz>
(Relative)
<a href = http://www.example.com/xyz>
(Absolute)

Data URLs

URLs prefixed with the data: scheme, allow content creators to embed small files inline in documents.

A few examples:

data:,Hello%2C%20World%21
The text/plain data Hello, World!. Note how the comma is percent-encoded as %2C, and the space character as %20.

data:text/plain;base64,SGVsbG8sIFdvcmxkIQ==
base64-encoded version of the above

data:text/html,%3Ch1%3EHello%2C%20World%21%3C%2Fh1%3E
An HTML document with <h1>Hello, World!</h1>

data:text/html,<script>alert('hi');</script>
An HTML document that executes a JavaScript alert. Note that the closing script tag is required.

MIME types

A media type (also known as a Multipurpose Internet Mail Extensions or MIME type) indicates the nature and format of a document, file, or assortment of bytes.

```
application/pdf
application/json
```

multipart/form-data

The multipart/form-data type can be used when sending the values of a completed HTML Form from browser to server.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types

https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types#multipartform-data

MIME types

MIME type	Audio or video type
`audio/wave` `audio/wav` `audio/x-wav` `audio/x-pn-wav`	An audio file in the WAVE container format. The PCM audio codec (WAVE codec "1") is often supported, but other codecs have limited support (if any).
`audio/webm`	An audio file in the WebM container format. Vorbis and Opus are the codecs officially supported by the WebM specification.
`video/webm`	A video file, possibly with audio, in the WebM container format. VP8 and VP9 are the most common video codecs; Vorbis and Opus the most common audio codecs.
`audio/ogg`	An audio file in the Ogg container format. Vorbis is the most common audio codec used in such a container; however, Opus is now supported by Ogg as well.
`video/ogg`	A video file, possibly with audio, in the Ogg container format. Theora is the usual video codec used within it; Vorbis is the usual audio codec, although Opus is becoming more common.
`application/ogg`	An audio or video file using the Ogg container format. Theora is the usual video codec used within it; Vorbis is the usual audio codec.

HTTP request methods

`GET`

The `GET` method requests a representation of the specified resource. Requests using `GET` should only retrieve data.

`HEAD`

The `HEAD` method asks for a response identical to a `GET` request, but without the response body.

`POST`

The `POST` method submits an entity to the specified resource, often causing a change in state or side effects on the server.

`PUT`

The `PUT` method replaces all current representations of the target resource with the request payload.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

HTTP request methods

`DELETE`

The `DELETE` method deletes the specified resource.

`CONNECT`

The `CONNECT` method establishes a tunnel to the server identified by the target resource.

`OPTIONS`

The `OPTIONS` method describes the communication options for the target resource.

`TRACE`

The `TRACE` method performs a message loop-back test along the path to the target resource.

`PATCH`

The `PATCH` method applies partial modifications to a resource.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

PUT VS POST

PUT	POST
Replacing existing resource or Creating if resource is not exist *http://www.example.com/customer/{id}* *http://www.example.com/customer/123/orders/456* Identifier is chosen by the client	Creating new resources(and subordinate resources) *http://www.example.com/ customer/* *http://www.example.com/customer/123/orders* Identifier is returned by server
Idempotent i.e. if you PUT a resource twice, it has no effect. Ex: Do it as many times as you want, the result will be same. x=1;	POST is neither safe nor idempotent. It is therefore recommended for non-idempotent resource requests. Ex: x++;
Works as specific	Works as abstractive
If you create or update a resource using PUT and then make that same call again, the resource is still there and still has the same state as it did with the first call.	Making two identical POST requests will most-likely result in two resources containing the same information.

Quiz

Can we use POST request to do all Request?

HTTP response status codes

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

Informational responses (100–199)
Successful responses (200–299)
Redirection messages (300–399)
Client error responses (400–499)
Server error responses (500–599)

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

User agent

A user agent is a computer program representing a person, for example, a browser in a Web context.

The user agent string can be accessed with JavaScript on the client side using the NavigatorID.userAgent property.

A typical user agent string looks like this: "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:35.0) Gecko/20100101 Firefox/35.0".

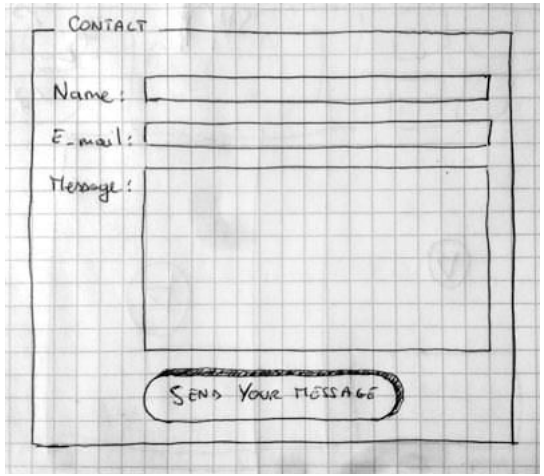
https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

HTML Elements

Tag	Description
<html> ... </html>	Declares the Web page to be written in HTML
<head> ... </head>	Delimits the page's head
<title> ... </title>	Defines the title (not displayed on the page)
<body> ... </body>	Delimits the page's body
<h *n*> ... </h*n*>	Delimits a level *n* heading
...	Set ... in boldface
...	Set ... in italics
...	Center ... on the page horizontally
...	Brackets an unordered (bulleted) list
...	Brackets a numbered list
...	Brackets an item in an ordered or numbered list
 	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a horizontal rule
<img src="...">	Displays an image here
 ... 	Defines a hyperlink

HTML Forms

Web forms — Working with user data




```
<form action="/my-handling-form-page"
method="post">
 <ul>
  <li>
    <label for="name">Name:</label>
    <input type="text" id="name" name="user_name">
  </li>
  <li>
    <label for="mail">E-mail:</label>
    <input type="email" id="mail" name="user_email">
  </li>
  <li>
    <label for="msg">Message:</label>
    <textarea id="msg"
name="user_message"></textarea>
  </li>
 </ul>
</form>
```

JSON Basics

Both JSON and XML can be used to receive data from a web server.

JSON stands for JavaScript Object Notation

```
'{"name":"John", "age":30, "car":null}'
```

a JavaScript program can easily convert JSON data into JavaScript objects.

- JSON stands for JavaScript Object Notation
- JSON is a lightweight data-interchange format
- JSON is plain text written in JavaScript object notation
- JSON is used to send data between computers
- JSON is language independent *

Data Types

In JSON, values must be one of the following data types:

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- *null*

JSON Basics

JSON Objects

```
{
"employee":{"name":"John",
"age":30, "city":"New York"}
}
```

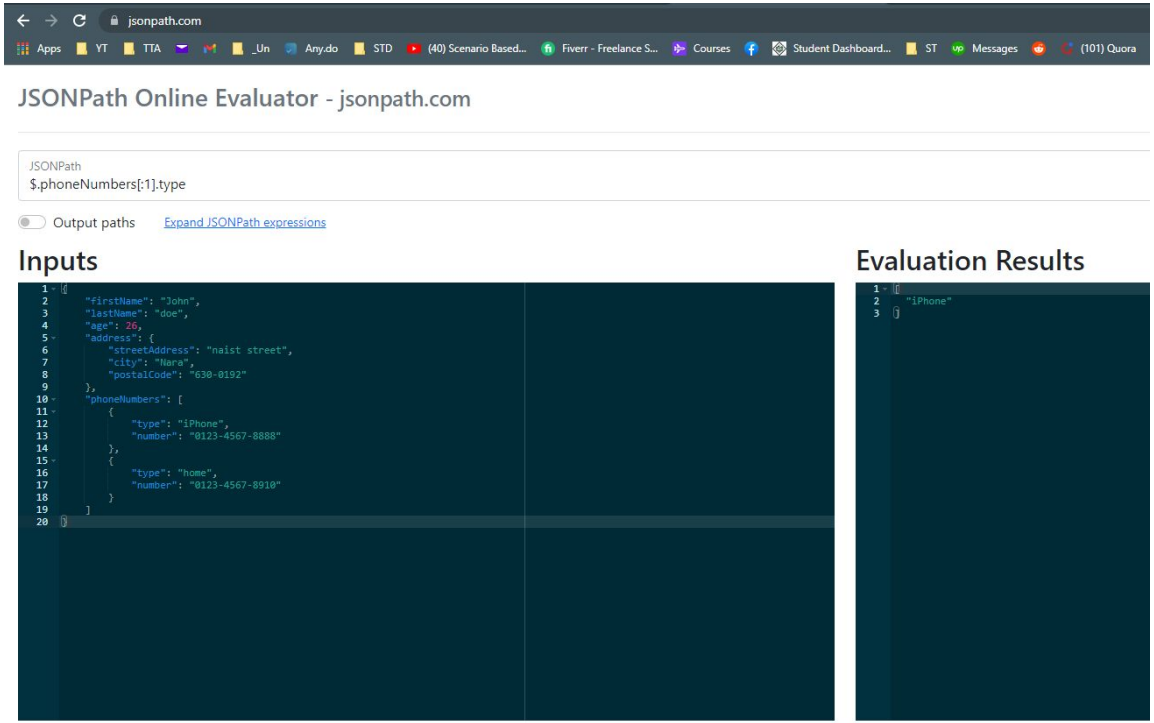
Objects as values in JSON must follow the JSON syntax.

JSON Arrays

```
{
"employees":["John", "Anna",
"Peter"]
}
```

JSON Path

https://www.jsonschema2pojo.org/



https://jsonutils.com/

https://jsonpath.com/

XML Basics

JSON Example

```
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```

XML Example

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

XML stands for eXtensible Markup Language.

XML was designed to store and transport data.

XML was designed to be both human- and machine-readable.

XML Basics

XML Example 2

```
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
<food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
   Two of our famous Belgian Waffles with plenty
of real maple syrup
   </description>
    <calories>650</calories>
</food>
```

https://www.w3schools.com/xml/tryit.asp?filename=tryajax_first

XML Basics

JSON is Like XML Because

- Both JSON and XML are "self describing" (human readable)
- Both JSON and XML are hierarchical (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XMLHttpRequest

JSON is Unlike XML Because

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

The biggest difference is:

XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

Why JSON is Better Than XML

XML is much more difficult to parse than JSON.
JSON is parsed into a ready-to-use JavaScript object.

For AJAX applications, JSON is faster and easier than XML:

Using XML

- Fetch an XML document
- Use the XML DOM to loop through the document
- Extract values and store in variables

Using JSON

- Fetch a JSON string
- JSON.Parse the JSON strin

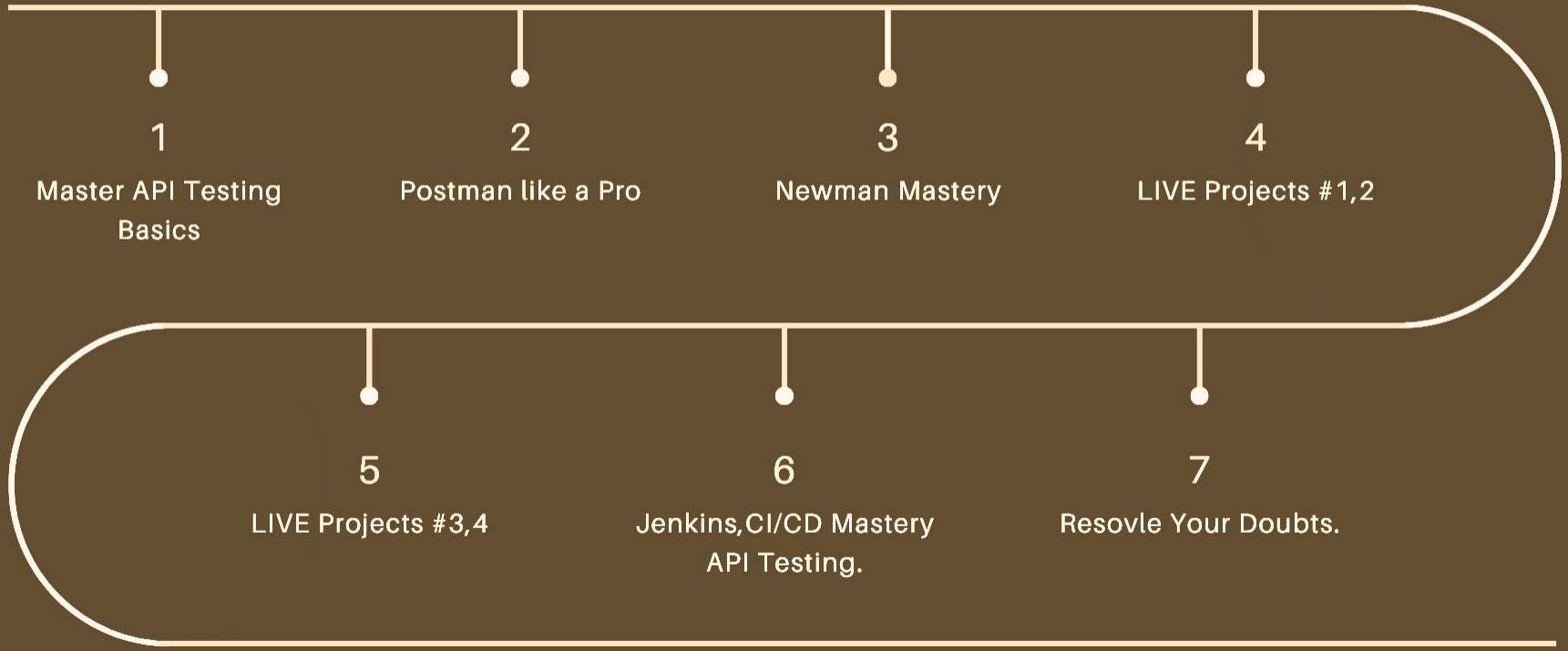
Install Tools for API Testing

Download Postman

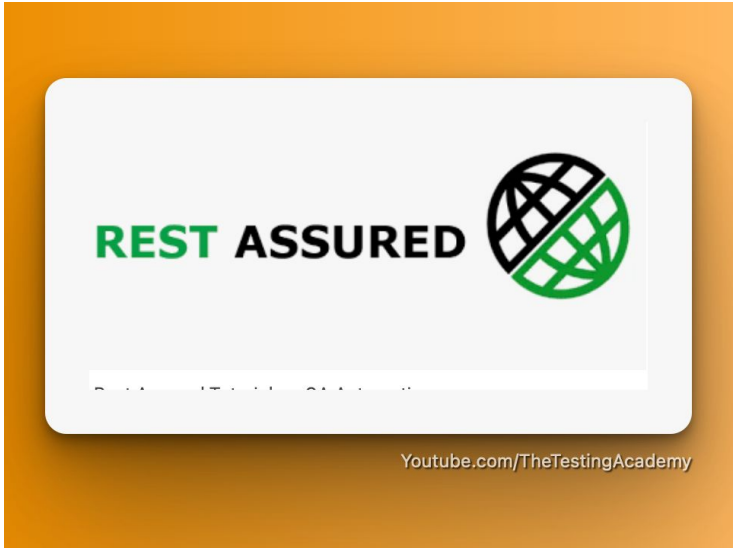
https://www.postman.com/downloads/

Mastering API Testing with Postman

LIVE Projects and Real Examples(With Worksheet, Checklists)



They are enough,



Chrome Dev Tools

Quiz

Type different HTTP Methods

Import Requests

Quiz

What is 404?

Quiz

Pick the status code class used for server error

1xx
2xxx
3xx
4xx

API Testing Tools

Quiz

What are the common protocols that are testing in API testing ?

HTTP
JMS
REST
SOAP
UDDI

How do You verify the GET ALL BOOKINGS

1. Status Code
2. Headers
3. Response Data
4. JSON Schema Validation
5. Data Types
6. Pattern, Regex
7. Invalid Auths/ Permission mismatch
8. Many others….

Thanks, for attending Class

I hope you liked it. Say Thanks in Comment :)

Fin.