MILESTONE #1

Getting started with REST APIs

REST APIs are those APIs which follow the guidelines set by the REST architecture. They follow a client-server model where one software program sends a request and the other responds with some data. REST APIs commonly use the HTTP protocol to send requests & receive responses.

How an API request differs from a usual HTTP request for a webpage is in terms of the data returned. HTTP requests for webpages return HTML, CSS & JavaScript files which are rendered by the browser and displayed to the user. But, in the case of APIs, the request can be for any data (not just webpage) and the response is read by the requesting program which interprets the data.

JSON is a standard format that is easily "understandable" by applications and can be handled well in most languages. So the data format in REST is usually JSON. For example, an Android app can effortlessly utilize data sent by a Node.js server. XML is another popular format for data transfer between applications.

The following video will give you a quick overview of REST API.
https://youtu.be/2-8CvFJ9Y4A

Trying it out

Excited about trying this out on your own? Use https://www.metaweather.com/api/location/search/?query=san or find one you like from here

Which ones did you try out? (You can try out a Chrome extension like JSONView to format the JSON response in your browser window)

As we discussed earlier, REST API calls are made on top of the HTTP protocol. We can analyse the network packets during the API calls to confirm this using Wireshark.

Wireshark is a popular network analysis tool to capture network packets and display them at a granular level. Once these packets are broken down, you can **use** them for real-time or offline analysis.

The client sends a HTTP **GET** request (line 8) to the server. The server responds with a **200** status code and JSON data on line 10. (API request was made to http://jsonplaceholder.typicode.com/posts, see [doc](doc))

No.	Source	Destination	Protocol	Source port	Dest Port	Info
5	192.168.5…	172.64.1…	TCP	48658	80	48658 → 80 [SYN] Seq=4111745307 Win=
6	172.64.13…	192.168.…	TCP	80	48658	80 → 48658 [SYN, ACK] Seq=3204376740
7	192.168.5…	172.64.1…	TCP	48658	80	48658 → 80 [ACK] Seq=4111745308 Ack=
8	192.168.5…	172.64.1…	HTTP	48658	80	GET /posts/1 HTTP/1.1
9	172.64.13…	192.168.…	TCP	80	48658	80 → 48658 [ACK] Seq=3204376741 Ack=
10	172.64.13…	192.168.…	HTTP	80	48658	HTTP/1.1 200 OK (application/json)
11	192.168.5…	172.64.1…	TCP	48658	80	48658 → 80 [ACK] Seq=4111745407 Ack=
12	192.168.5…	172.64.1…	TCP	48658	80	48658 → 80 [FIN, ACK] Seq=411174540

```
> Frame 10: 1137 bytes on wire (9096 bits), 1137 bytes captured (9096 bits)
> Ethernet II, Src: 0a:94:23:ac:62:2c (0a:94:23:ac:62:2c), Dst: 0a:b1:30:3f:db:88 (0a
> Internet Protocol Version 4, Src: 172.64.132.18, Dst: 192.168.58.123
> Transmission Control Protocol, Src Port: 80, Dst Port: 48658, Seq: 3204376741, Ack:
> Hypertext Transfer Protocol
v JavaScript Object Notation: application/json
  v Object
    v Member Key: userId
        Number value: 1
        Key: userId
    v Member Key: id
        Number value: 1
        Key: id
    v Member Key: title
        String value: sunt aut facere repellat provident occaecati excepturi optio re
        Key: title
    v Member Key: body
        String value: quia et suscipit\nsuscipit recusandae consequuntur expedita et
        Key: body
```

References

1. Understanding & using REST APIs
2. HTTP vs REST
3. REST Guidelines
4. Metaweather API Documentation
5. Installing Wireshark on Ubuntu
6. Capturing HTTP network packets using Wireshark

Curious Cats

- Are the API endpoints case sensitive i.e, if requests to /location & /Location must return the same response? Try it out for https://www.metaweather.com/api/location/search/?query=san
- Why is it that you are able to make a REST API call via the browser?

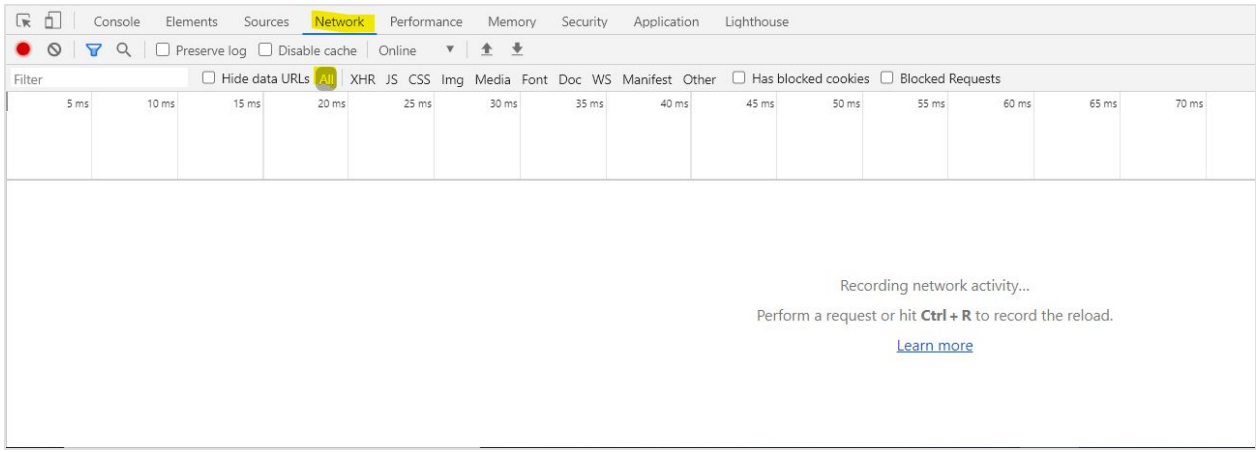
MILESTONE #2

Use Chrome Developer Tools to understand HTTP structure

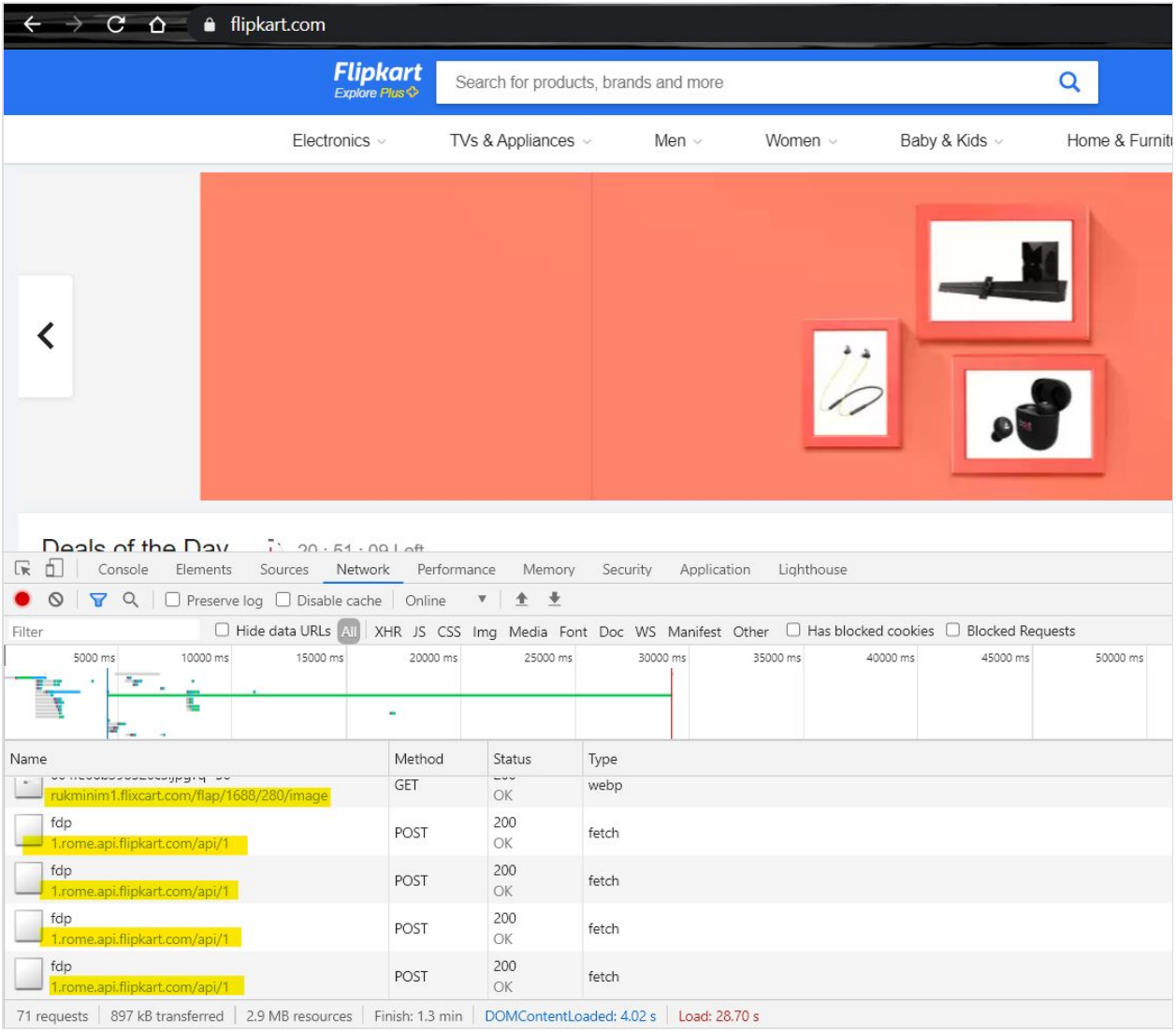
If you already know how to use Chrome Developer Tools, you can skip this section and move on to the next section.

Chrome browser provides inbuilt tools to peek into the HTTP traffic it makes. This information can be used to better understand what's happening behind the scenes when some URL is visited or an action like clicking the *Login* button is performed.

1. To open Chrome Developer Tools, press `Ctrl + Shift + i` / `Cmd + Shift + i` in the browser window and select the `Network` tab.

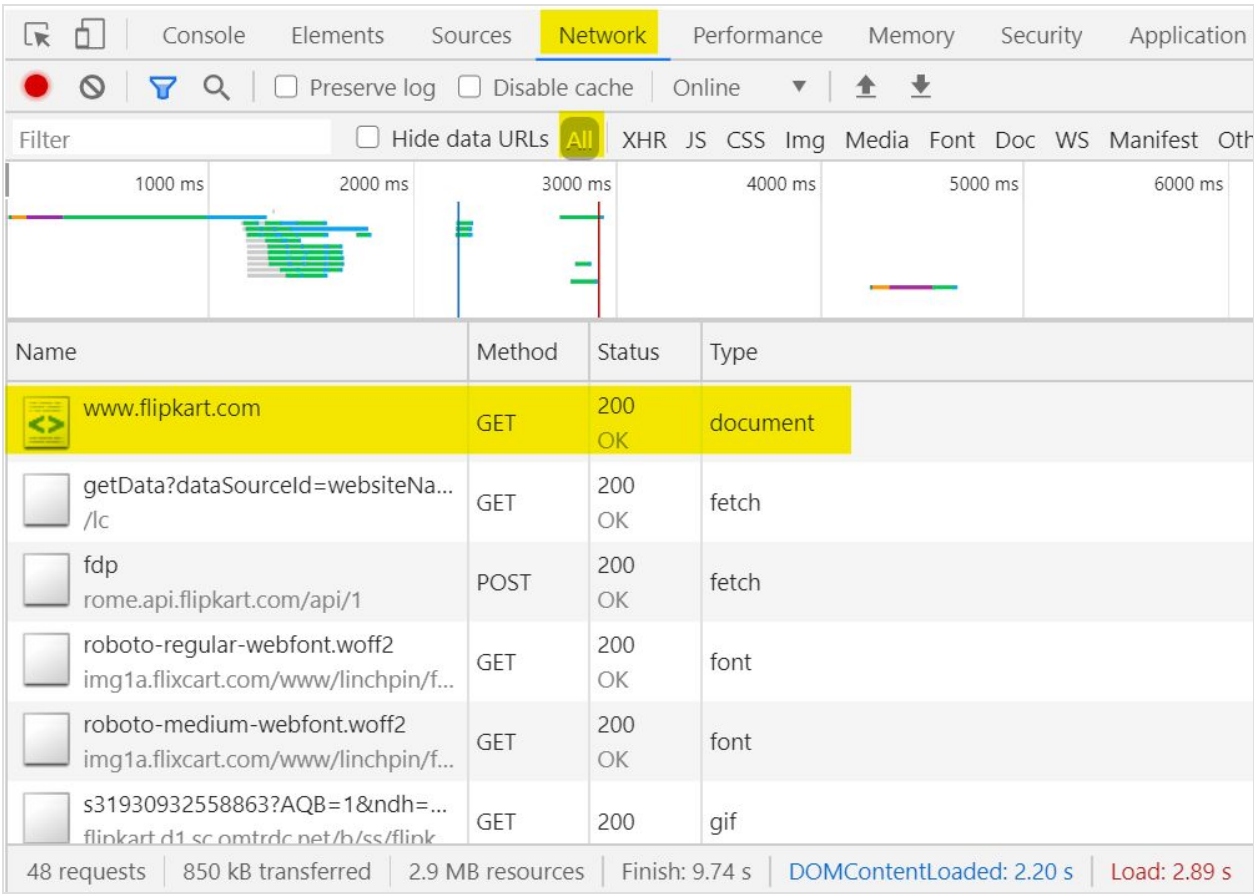


2. Try visiting a website (eg: https://www.flipkart.com/) to see the HTTP requests getting populated. You will see many HTTP requests being made.

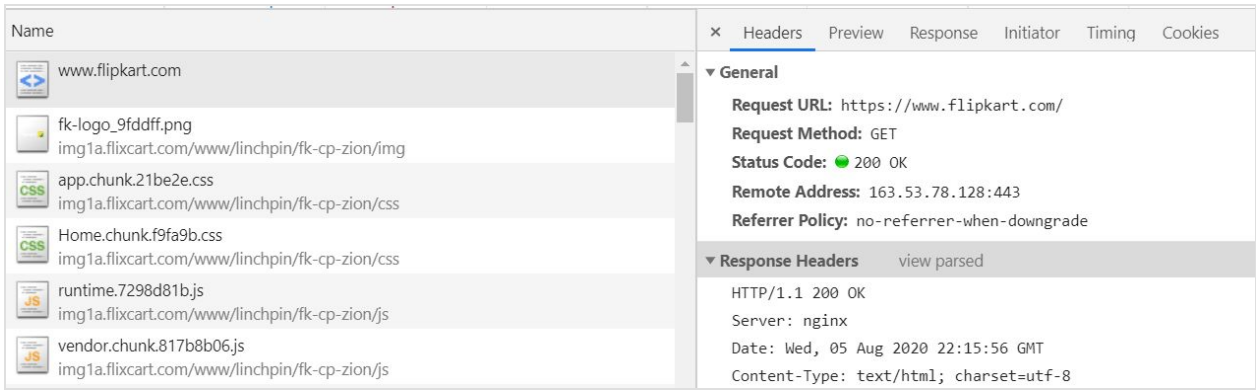


3. Scroll to the top of the network activity to find a HTTP **GET** request to **www.flipkart.com**. (Find the entry In the `Name` tab, you should see

`www.flipkart.com` with `Type` as **document**).



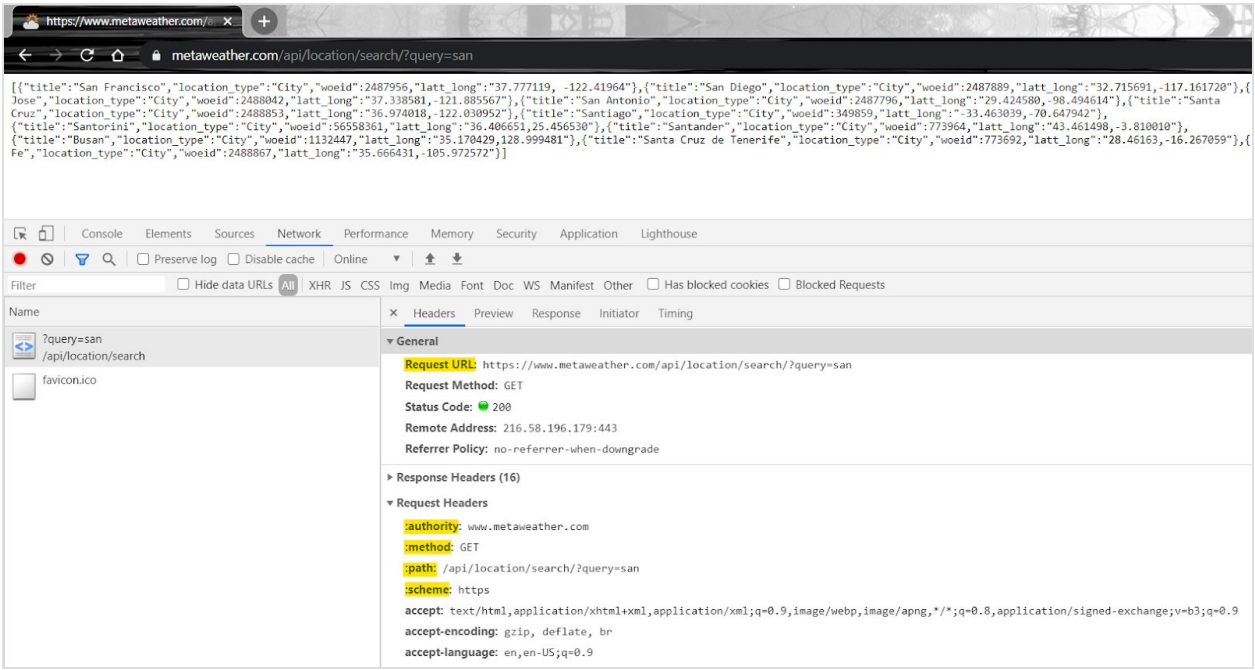
4. Click on the entry to open a side-bar with information regarding the request & response for it



Components of a REST API Request

Use Chrome's Developer Tools to monitor the API request made. Let's look at the different components of the REST API request.

1. Request URL: https://www.metaweather.com/api/location/search/?query=san
2. Request Method: **GET** which denotes the type of HTTP request made. GET means data needs to be fetched.
3. Request Headers: eg: **accept**, **accept-encoding** - used to send additional info like the type of encoding that the requesting application (browser) supports
4. Request Body: is empty for the current request but can be used for sending additional information like a file's content when uploading it to the server.



The Request URL is made up of the

1. Scheme: **https** - denotes the request was made using the HTTPS protocol ie, secure version of the HTTP protocol
2. Root-endpoint: **www.metaweather.com** - defines the API provider

3. Path: **/api/location/search/** - there will be one api path for each type of resource. Here, we are asking for the resource named *location*.

Location Search

URL

`/api/location/search/?query=(query)` `/api/location/search/?lattlong=(latt),(long)`

Arguments

Either query or lattlong need to be present.

query
Text to search for.
lattlong
Coordinates to search for locations near. Comma separated lattitude and longitude e.g. "36.96,-122.02".

Examples

- /api/location/search/?query=san
- /api/location/search/?query=london
- /api/location/search/?lattlong=36.96,-122.02
- /api/location/search/?lattlong=50.068,-5.316

4. Query parameter: **?query=san** - the part of the URL that comes after a **?** character is the query parameter. It specifies the search criteria for the resource. Here, the locations returned get filtered by the value of the query parameter, **query** we provide.

For every API request, the corresponding API response also contains HTTP headers that the server sends back along with the data requested. See if you can answer some questions based on these response headers.

▾ Response Headers

```
allow: GET, HEAD, OPTIONS
cache-control: private
content-encoding: gzip
content-language: en
content-length: 374
content-type: application/json
date: Fri, 14 Aug 2020 06:20:23 GMT
server: Google Frontend
status: 200
strict-transport-security: max-age=2592000; includeSubDomains
vary: Accept-Language, Cookie
vary: Accept-Encoding
x-cloud-trace-context: 3bad9d414311dd96fa2b0bf473569a6b
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 1; mode=block
```

- What are the HTTP methods this API endpoint supports?
- What is the data format sent by the server?
- Check the response encoding used. Was it included among the **accept-encoding** request header sent to the server?

MILESTONE #3

REST API calls using Programs

We saw how to make REST API calls using the browser. But, what was the purpose of having an API? It was for applications to communicate with each other, right? Let's now see how to do that programmatically.

Use `curl` on your command line to make a REST API call to https://www.metaweather.com/api/location/search/?query=san. This fetches location information for locations matching the **query** parameter *san*

```
crio-user@crio-demo:$ curl https://www.metaweather.com/api/location/search/?query=san
[{"title":"San Francisco","location_type":"City","woeid":2487956,"latt_long":"37.777119, -122.41964"},{"title":"Sa
:"City","woeid":2487889,"latt_long":"32.715691,-117.161720"},{"title":"San Jose","location_type":"City","woeid":24
8581,-121.885567"},{"title":"San Antonio","location_type":"City","woeid":2487796,"latt_long":"29.424580,-98.494614
,"location_type":"City","woeid":2488853,"latt_long":"36.974018,-122.030952"},{"title":"Santiago","location_type":"
tt_long":"-33.463039,-70.647942"},{"title":"Santorini","location_type":"City","woeid":56558361,"latt_long":"36.406
:"Santander","location_type":"City","woeid":773964,"latt_long":"43.461498,-3.810010"},{"title":"Busan","location_t
447,"latt_long":"35.170429,128.999481"},{"title":"Santa Cruz de Tenerife","location_type":"City","woeid":773692,"l
crio-user@crio-demo:$
```

Let's see how to do the same using a Python program

```python
# Import a library that allows to make HTTP request
import requests

# Set the API endpoint
url = "https://www.metaweather.com/api/location/search/?query=san"

# Use the library to perform an HTTP GET request to the URL
response = requests.get(url)

# Print out the data
print(response.text)
```

Try the program out and see if you get a similar response to that with `curl`. You can use this online Python client to run the code.

Java program to do the same

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
```

```java
import java.net.MalformedURLException;

import java.net.URL;

public class Main {

	public static void main(String[] args) throws MalformedURLException, IOException {

		// create url

		URL url = new URL("https://www.metaweather.com/api/location/search/?query=san");

		// Send Get request and fetch data

		HttpURLConnection conn = (HttpURLConnection) url.openConnection();

		conn.setRequestMethod("GET");

		BufferedReader br = new BufferedReader(new InputStreamReader(

			(conn.getInputStream())));


		// Read data line-by-line from buffer & print it out

		String output;

		while ((output = br.readLine()) != null) {

			System.out.println(output);

		}


		conn.disconnect();

	}

}
```

You can use [this](#) online Java client to try the code and play around making changes. Spring framework provides a [RestTemplate](#) class if you don't want to deal with lower level details like opening a connection & buffers.

Do the below tasks

- Use the above API endpoint to find the value `<woeid>` attribute for your city (if you get an empty response for your city, try *bangalore* :))

(A **WOEID** (Where On Earth IDentifier) is a unique 32-bit reference identifier that identifies any feature on Earth)

- Use the **woeid** value you found to call this API endpoint - replace "(woeid)" with your city's woeid value - https://www.metaweather.com/api/location/(woeid)/
- In the response, the **weather_state_name** attribute denotes what the weather will be like *Light Rain*, *Clear* etc on a particular day denoted by the **applicable_date** parameter. Find the weather for today.
- **Challenge Task** - Write a program to print out the weather for today using the above API endpoints

References

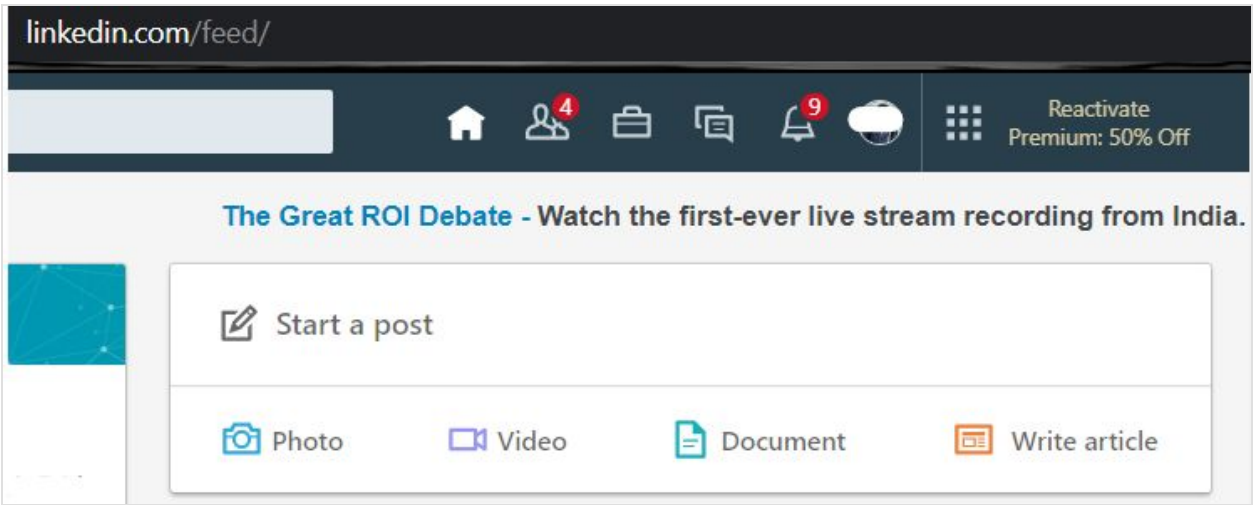
1. Metaweather API Documentation
2. HTTP requests using Java - HTTPUrlConnection
3. HTTP requests using Java - Spring RestTemplate
4. HTTP requests using Python
5. Idempotency in REST APIs

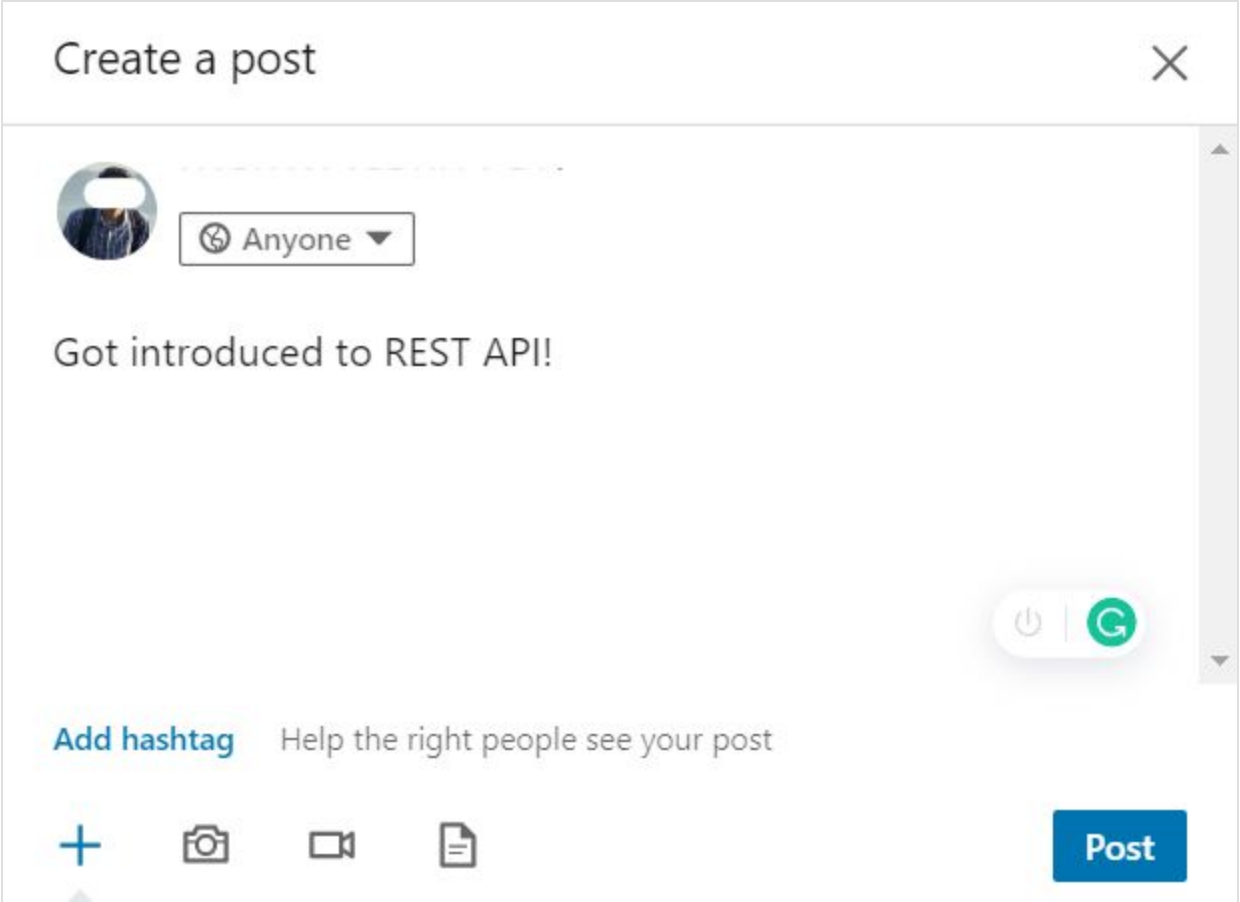
MILESTONE #4

LinkedIn with REST API

Congrats! REST APIs aren't a mystery anymore. Why don't you let the world know about it? Share a post on LinkedIn - "Got introduced to REST API!".

I'll lead by example :). Goto https://www.linkedin.com/feed/ and add your post message.



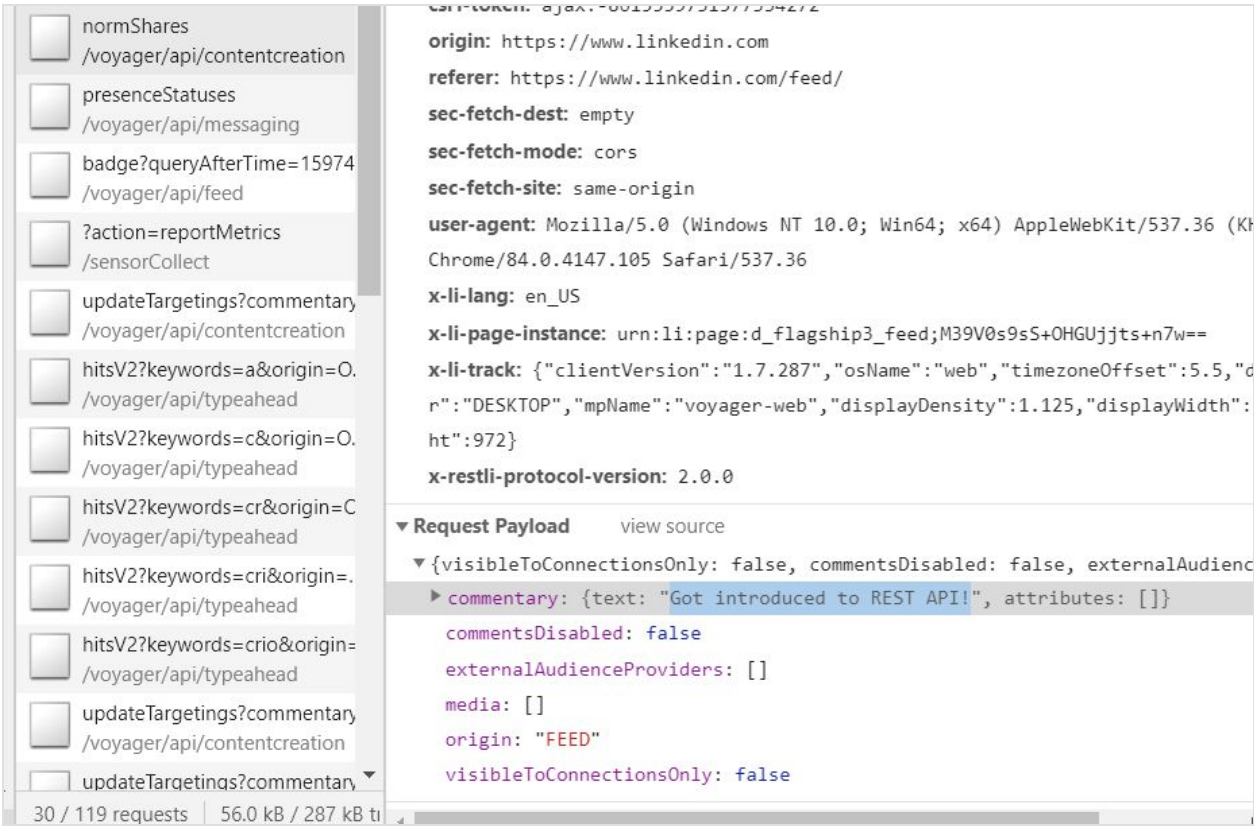


Wait! Now that we know most of the communication these days are via REST APIs, why don't we keep the Chrome Developer Tools window open when we click the **Post** button?

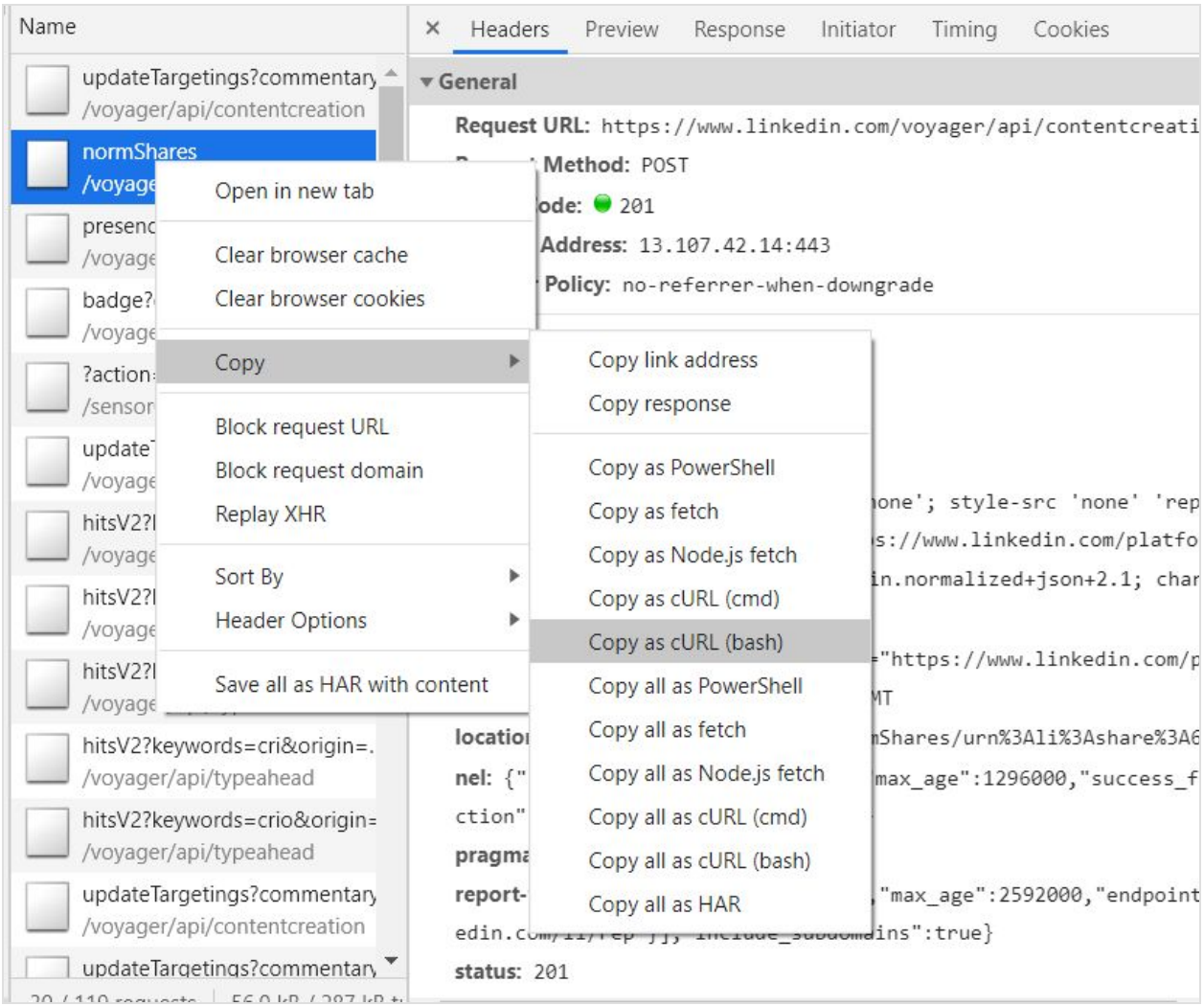
An HTTP POST request to https://www.linkedin.com/voyager/api/contentcreation/normShares happened when the **Post** button was clicked. This could be the API endpoint that LinkedIn used to post our message.

Name	Method	Status	Type	Waterfall
track /li	POST	200	tex...	
gbtn-progress.svg kbfnbcaeplbcioakkpcpgfkobkghlhen/src/images/301b547f217d5d2ee02...	GET	200 OK	svg...	
updateTargetings?commentary=Got%20introduced%20to%...cleAndText... /voyager/api/contentcreation	GET	200	xhr	
track /li	POST	200	tex...	
track /li	POST	200	tex...	
b?c1=2&c2=6402952&c3=&c4=&c5=&c6=&c15=&ns__t=15974...8&c... sb.scorecardresearch.com	GET	204 No Content	tex...	
track /li	POST	200	tex...	
gbtn-disable-icon.svg kbfnbcaeplbcioakkpcpgfkobkghlhen/src/images/765d0f8abb3e637eff82...	GET	200 OK	svg...	
normShares /voyager/api/contentcreation	POST	201	xhr	
presenceStatuses /voyager/api/messaging	POST	200	xhr	
track /li	POST	200	tex...	
badge?queryAfterTime=1597411071614&countFrom=0 /voyager/api/feed	GET	200	xhr	

Click on the request entry to open it & see if you can find the post message in the **Request Payload** section in the **Headers** tab

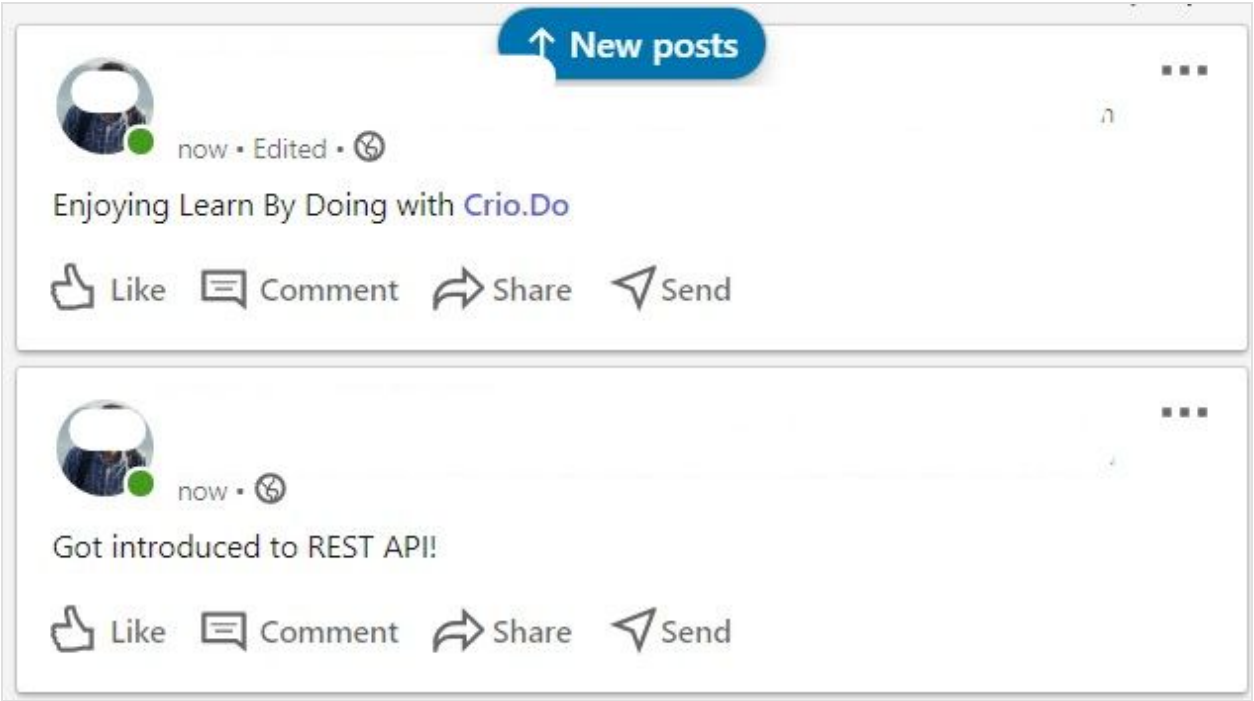


Yep, it's there. Let's try to use the `curl` command to perform the same action. DevTools will give you that out of the box!



Before you get all excited and execute the `curl` command copied from DevTools, take a step back and think what would happen if you do that - You'll again post the same message. No fun, right?

Copy-paste the `curl` command to a text file, search for your post message (it'll be part of the **--data-binary** flag), and change it to something else. Try executing the command from your terminal now, are you able to see this message posted? (There's a **text** parameter with value **Got introduced to REST API** for me. I updated it to **Enjoying Learn By Doing with Crio.Do** for the new post)



But, how did LinkedIn know whose account to post the message to?

If you observe carefully, you will see cookies from the browser get sent as a part of the request. This is what LinkedIn uses to identify your account. This cookie gets refreshed periodically, so it may not remain valid forever. A new one would be generated after a while.

We saw how to post messages using the REST API LinkedIn provided. Why don't you try the same with Youtube? See if you can find out the API endpoint used to search for videos.

Pretty cool, right? Go wild, doing cool things with REST API!

Summary

- We have learnt what REST API is by running some of the APIs.
- APIs makes it easier for
    - Applications to expose their services
    - Other applications to avail those services.

    - Integration is easier, only the API definitions need to be exposed.
- The internet is full of such APIs and the knowledge of APIs will help you utilize these services as well as to develop new ones of your own
- Find pointers to the Curious Cats questions here
- Further Reading
    - Generating Code Snippets using Postman
    - Right way to design REST URL Best Practices 1 and Best Practices 2
    - 5 Basic REST API Design Guidelines
    - REST vs GraphQL APIs, the Good, the Bad, the Ugly
    - SOAP vs REST

Newfound Superpowers

- Know-how of REST APIs

Now you can

- Make REST API calls
- Explain how REST API calls differ from normal HTTP requests for web pages