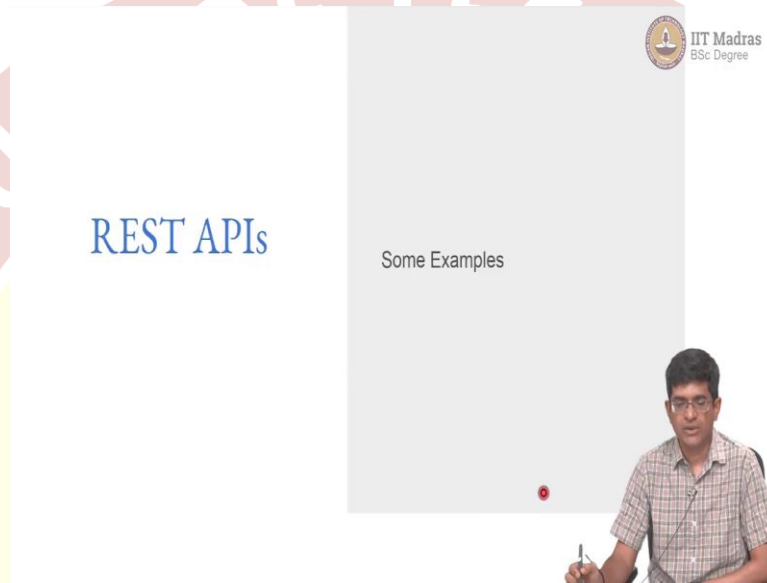


**IIT Madras**  
ONLINE DEGREE

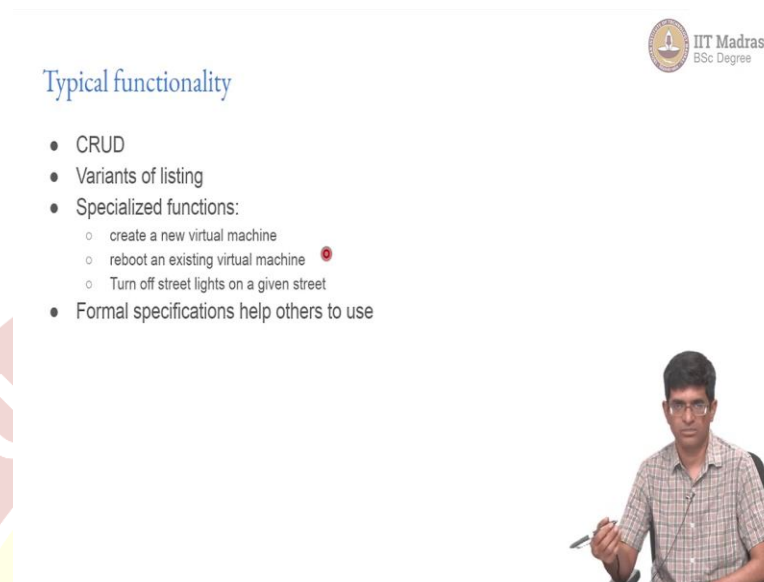
**Modern Application Development - I**  
**Professor Nitin Chandrachoodan**  
**Department of Electrical Engineering**  
**Indian Institute of Technology, Madras**  
**REST APIs - Examples-I**

(Refer Slide Time: 0:17)



Hello, everyone, and welcome to this course on modern application development. So, now that we have a reasonably clear understanding of what REST stands for, let us look at some examples to make it a bit more concrete. So, far, it is been a little abstract, we have been talking about notions of software architecture, and sort of motivating the idea of why you need to transfer state around and so on. But what exactly does that mean? And how does it actually get implemented in practice? Let us look at some concrete examples.

(Refer Slide Time: 0:42)



Typical functionality

- CRUD
- Variants of listing
- Specialized functions:
  - create a new virtual machine
  - reboot an existing virtual machine
  - Turn off street lights on a given street
- Formal specifications help others to use

IIT Madras  
BSc Degree

So, the typical functionality that you would expect to do using these, this REST kind of approach is one is CRUD, the Create, Read, Update, Delete, which is basically data management. But, the read part of it could have many variants, you might want to sort of display things in a page as a list, you might want to sort of get a histogram of data, so the question is, where does REST really play a part in this, it is the way by which the data is transferred between the model and the controller.

And the controller by choosing the appropriate view, can then present things to you in a different way. So, that is where basically, the REST is coming into the picture. Now, the interesting thing about REST is it is not limited just to data management. And in particular, you will find that in the Google Cloud, the Google Cloud Console, for example, they have APIs of their own, that, for example, allow you to create a new virtual machine, reboot an existing virtual machine. And you could even have things for example, let us say you are trying to build, a smart city application, you might want to have your server, which controls the lights around the city having a restful API.

So, what does that mean? It basically means that it exposes an API of its own, one element of the API might be check the state of the lights on the street, and other element might be turn on the light on the street, or turn off the light on that street. And so, in other words, that by itself, any of those commands is not going to go in and change any kind of underlying data model, you might

have a database, which stores the state of the light somewhere, but it is not necessary. I mean, I might, just have something like reboot a machine, reboot a machine is not really a task, which is, doing one of the CRUD operations, it is actually doing something else to have, an actual virtual machine somewhere.

And the important point about all this is, you have all of this functionality, and you need some way by which you can formally specify it in such a way that it is useful to others, other people could actually read through this and make use of it at some point.

(Refer Slide Time: 2:55)

The slide is titled "Example: Wikipedia" in blue text. It features a bulleted list of four items: "Open API", "Search for pages", "History of page", and "JSON output". In the top right corner, there is a logo for "IIT Madras BSc Degree". In the bottom right corner, there is a small video inset showing a man with glasses and a plaid shirt, who appears to be the speaker. The entire slide is overlaid on a large, semi-transparent watermark of the IIT Madras logo, which includes the text "INDIAN INSTITUTE OF TECHNOLOGY MADRAS" and the Sanskrit motto "विद्यिर्भवति कमलजः" (Vidyirbhavati Kamalaja).

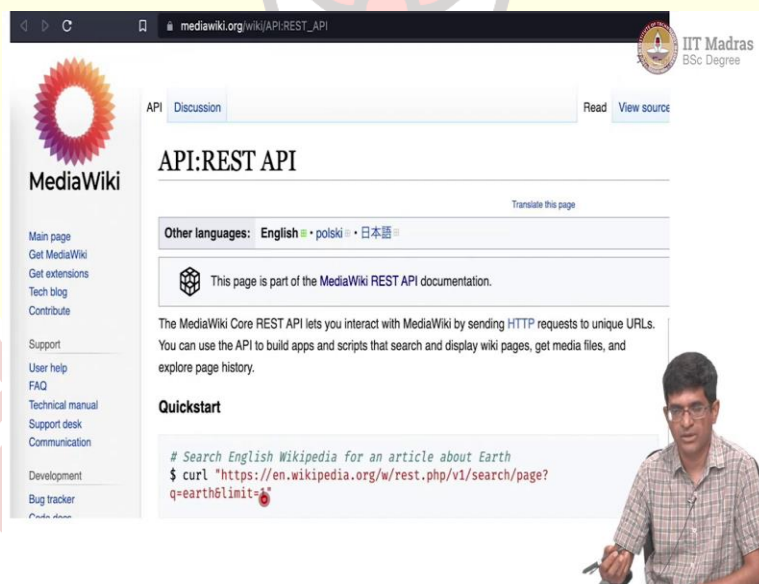
So, I am going to start by taking one specific, we will be looking at two examples. The first one is Wikipedia. Now, we all know what Wikipedia is, it is basically sort of an online encyclopedia, in principle, anyone can go and edit any page on it, in practice, they try not to, at least the more important pages, they put some kind of protection on it so that to avoid possibilities of abuse, but it is still remarkably free, you can go in and pretty much change any page that you want, and provide updates on it.

And in some cases, people will, look at it and say, yeah, this looks good, and accept those changes into Wikipedia. Now, what does the API do? Why do you need an API for such a thing after all I have, this webpage out there, I have a search box, I can go search for anything, and bring it up. Now, people might want to do a bit more than that. So, for example, they may want to say, for certain kinds of search terms, can I find how many pages are there.

Or is there a way by which I can find out the total number of pages in a particular language? Or can I do something like searching for everything, which starts with the letter ABC, so, of course, if I have access to the database, I can do all of that. But if I do not have access to the database, then it becomes quite painful to actually, physically go there and search for each and every one of those.

So, what the Wikipedia founders did was they created an API, which allows you to do that you can search for things like, search for a term, and it will return all pages that contain that term, mostly in the title or something like that. You can retrieve the history of a page, what that means is you can basically go in and say, when was this edited, who edited it? When was it last done, rather than sort of going and reading all of that you can have a program that sort of pulls it out for you and maybe presents it to you in a different way. So, all of this, just like I was describing earlier happens in the form of JSON output that is going to come when you make requests to certain URLs.

(Refer Slide Time: 5:01)



So, this is the main page that describes the REST API corresponding to Wikipedia, you can see that the location of this, I mean, you just Google Wikipedia API, and it pretty much takes you here. Media Wiki is sort of the parent organization that handles Wikipedia plus certain other things. It is more like the entire mechanism that is used for implementing Wikipedia. And as you

can see over here, there is a lot of information on it. Of course, it says bold up, up at the top that this is the REST API for Wikipedia.

It also gives you information, a quick start, how do you, for example, search the English Wikipedia for an article about earth? Now, look at what is happening. Basically, what it is saying is, you have this URL `en.wikipedia.org`, that tells it that it is the English version of Wikipedia. And after that comes this `w/rest.php`. That is the part that basically says, it is taking you to the REST API, rather than to something which will actually give you back the HTML page.

And the fact that you use this `rest.php` over here has implications on the kind of output that you get from the system, finally, you will get JSON output rather than HTML. It also has one `v1` out here and this is something you will commonly find in most REST APIs. It is the API version. Why is that important? Because one of as we will see later, you might have a scenario where you need to change something in an API.

And there is something fundamental about the notion of an API, which basically says that you cannot just break what another person is expecting will work. So, if something that you are going to change is going to break what someone else expects to work, you need to make that clear, and ideally, make it a different version, a new version of whatever you are trying to do. So, anyway, this is the `v1` tells you the version of the API.

And what kind of resource are you doing? You are basically performing a search on a page. So, if you ask what is the exact resource over here this is sort of, this `search/page` is the resource that you are activating at this point. So, it is not a page by itself, that is the resource over here. It has this functionality that is being activated at this point, what is being done is you are passing parameters to it, there is a `q`, which is the query string. And something else `limit` equal to 1 is just something which some additional information which is used by the API.



(Refer Slide Time: 7:50)



curl "https://en.wikipedia.org/w/rest.php/v1/search/page?q=earth&limit=1"

```
{
  "pages": [
    {
      "id": 9228,
      "key": "Earth",
      "title": "Earth",
      "excerpt": "<span class='searchmatch'>Earth</span> is the third planet from the Sun and the only astronomical object known to harbor and support life. About 29.2% of <span class='searchmatch'>Earth</span>'s surface is land consisting",
      "description": "Third planet from the Sun in the Solar System",
      "thumbnail": {
        "mimetype": "image/jpeg",
        "size": null,
        "width": 200,
        "height": 200,
        "duration": null,
        "url": "//upload.wikimedia.org/wikipedia/commons/thumb/9/97/The_Earth_seen_from_Apollo_17.jpg/200px-The_Earth_seen_from_Apollo_17.jpg"
      }
    }
  ]
}
```

So, what do you think will happen if you actually execute this? Well, we can try it out. The simple way to do it is just run this command curl. Now, for the purpose of this presentation, I have run all of these commands ahead of time. And I am only pasting the data out here. But I would strongly encourage all of you to try this out on your own. Make sure that you try it, see what happens, make changes to it, search for something else, change this limit over here, see what happens if you change to another language, try anything you want, be careful while trying it. Because if you sort of put it in a script that generates too many requests, there is a good chance that Wikipedia will find that to be abusive, it is not a good use of their API. And they will blacklist you, they will try to block your machine from connecting to it.

That is true of any API. The people who are providing it or providing it as a public service, so that you can do something useful to it. Always keep that in mind, when you are trying to connect to an API, play nice, limit the number of requests that you use, over there and sort of stagger them out over a longish time. So, that it does not overload their servers and does not sort of cause them to have to say, look, you are causing problems for us, we need to block you.

Now, anyway, the interesting thing here is curl is something that we have mentioned earlier, it is just a command line, URL retriever. And by command line, what I mean is this is ideally, if you are on Linux system, curl should be pretty much in most cases already installed on your system. If not, you can install it very easily. Even if you are on Mac OS, I believe call is typically there

by default, on Windows you might have to install it separately, it is, I believe, also there in the shell on Replit and you should be able to sort of try it on that also.

All that it is doing is it takes a URL, in this case, an HTTPS URL, connects to this particular server and sends it this request. So, this is the query string, so /w/rest.php, etc. is the actual URL query string, out of that this q equal to blah, blah, blah is the specific portion of the string, which is the query to be passed into this part of the application. So, curl will just initiate an HTTPS request, it will connect to port number 443, on whatever IP address corresponds to this machine and say, get, because curl by default will send GET requests. So, HTTP GET request will be sent for this.

What will we see as the response? So, like I said, I already tried this out. And I am just going to show you what the response is, in this case it comes back with a JSON object, you can see that there are like curly brackets, starting here, ending here. And it is again, a key value, it is a dictionary, just like the person identifier that I showed earlier. The data in it is, first you know you have the pages.

So, it basically says pages, so you could have multiple pages coming back for earth. But since we are limited to one, it is only giving back the first one. The id and this pages is basically an array, you can see that because it starts and ends with curly brackets, or rather square brackets. Within that, you can see that there is a specific object, which starts with these curly brackets out here and ends with these curly brackets. That object has an id 9228 in this case, the key for that object is the actual title Earth itself, or rather, the value Earth, the title of that page is Earth.

And it also provides you a small excerpt from the page, if you look at it, this part is straight HTML. It is just dumping this out. And you can see that it is literally dumping it out, because it ends in the middle of a sentence, is land consisting, just stops there, it basically picked up so many words from the page and sent it out to you. So, that is an excerpt, it basically has pulled out the first few words from the HTML and sent it.

There is also a brief description. This is one of the things that so called metadata about the page, which would not show up on the page as such, but it is provided by the authors as part of the thing, basically says, description third planet from the Sun and the solar system. You might not



see that on the page itself. But if you look at the page source, you will probably find there is a description tag, which says, this is what it is.

It also provides a link to a thumbnail a picture, the Earth seen from Apollo 17, and so on. So, what is the bottom line? Because we made a request to the REST API, as specified by this, we gave it a certain query string. And we just used curl, which means it is an HTTP GET request. This is the information we got back in response. This JSON blob, a blob is basically just a chunk of data. So, this is useful, it means that by changing Earth to something else, I could continue searching for various other things on Wikipedia. You might want to change limit equal to something else and see whether you get multiple pages corresponding to Earth and what do they each correspond to.

Thing is if you do this from inside a Python program, for example, this would straightaway coming to you as a JSON blob, which can be easily passed within Python and you would get a dictionary which contains this, you can then run through the dictionary and say, these are the number of objects and so on, like, all of that is very nicely in place for you.

(Refer Slide Time: 13:30)



```
https://en.wikipedia.org/w/rest.php/v1/page/Main_Page/history

{"revisions":[
{"id":1004593520,"timestamp":"2021-02-03T11:11:30Z","minor":
false,
  "size":3508,"comment":"cut",
  "user":{"id":2927383,"name":"Izno"},"delta":28},
{"id":1004592788,"timestamp":"2021-02-03T11:03:39Z","minor":
false,
  "size":3480,"comment":"cut"
. . .
```

Like I said, another thing you could do is, for example, pick up the history of a page. So, what is history? In this case? It is how often has the page been revised? What has happened to it, what has been changed over time, and so on. Now, this, obviously, this is the main page of Wikipedia, it undergoes quite a lot of revisions, I mean, there are like small changes that keep getting

(( ))(13:53) every now and then. So, when I look at it, I find that there are, it comes back with an array of revisions, .

And I have not even put in the whole lot, because quite a lot of data comes back. Not a whole lot because it I think limits the number of revisions that it is showing to you. But it basically shows, what information does that contain each of those objects, basically, it starts with an id and goes up to this point. And other one, it starts with an id and goes up to this point, and so on. What does it have, it has a timestamp on which it was updated. It has a Boolean flag, which indicates whether it was a minor change or not. The size of the change, how many, I believe this is the final number of bytes or something.

And this comment, I mean, it is not actually cut I just cut this out over here for ease of display. It is actually quite long, it basically gives some description of what was changed. It also says who was the person who changed it, what were the number of changes and so on. So, you can easily pull this out. This for example will easily tell you will, you can write a small script around this which, give it any page. And it will tell you how many times it changed. How often was it changed, when was the last change made, all of that information can be collected easily.

There are a lot more things in the API, you could for example, go by user, you can find out which users have been contributing the most. So, the fact that it provides an API like this makes all of this possible. Otherwise, you would need to sort of sit and download each of the pages of Wikipedia one by one, parse the HTML, try to get some data from there, and hopefully not make a mistake. Whereas over here, you are straightaway able to hit the database at its back end and get useful information from it. As I said, this is a public service, Wikipedia was not forced to do this, they are providing it because it is useful for many people. So, keep that in mind when you access APIs in general.

(Refer Slide Time: 15:51)



## Documentation

Schema [edit]

id	Page identifier
required   integer	
key	Page title in URL-friendly format
required   string	
title	Page title in reading-friendly format
required   string	
excerpt	<p>For search pages endpoint: A few lines giving a sample of page content with search terms highlighted with <code>&lt;span class=\`searchmatch\`&gt;</code> tags</p> <p>For autocomplete page title endpoint: Page title in reading-friendly format</p>
required   string	
description	Short summary of the page topic based on the corresponding entry on <a href="#">Wikidata</a> or <code>null</code> if no entry exists
required   string	

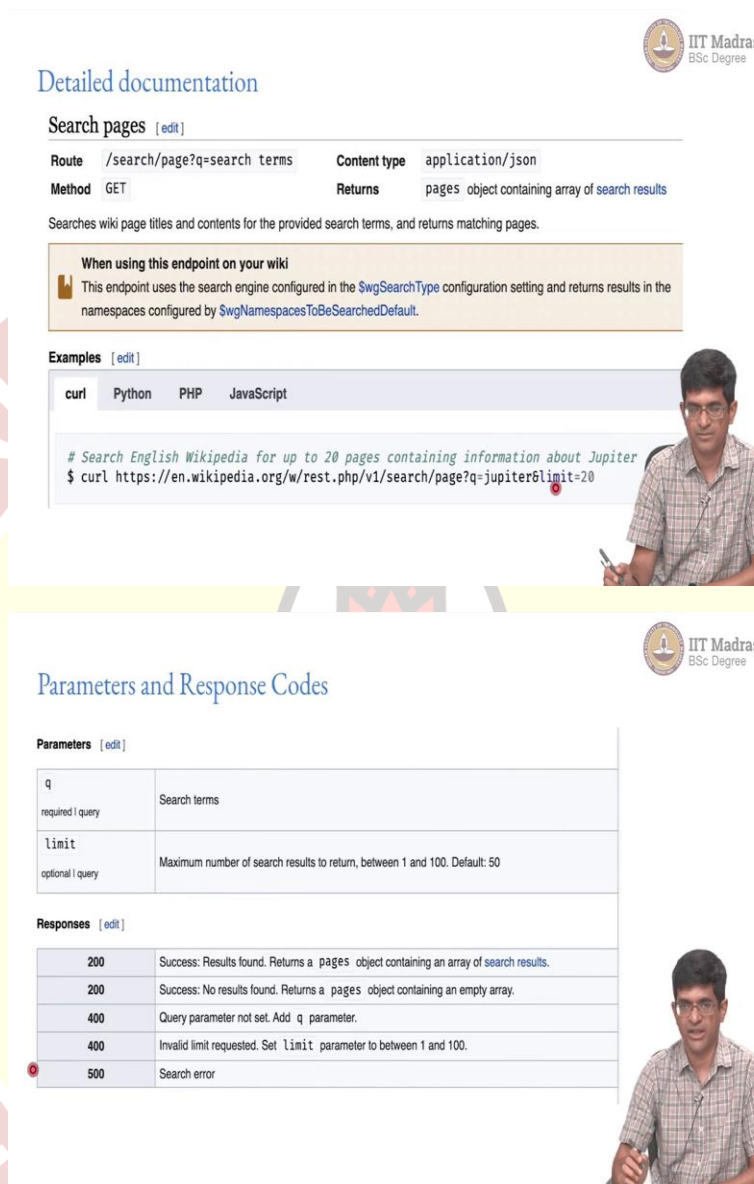


Now, a bit about the documentation provided for this, it provides something called a schema, which basically tells you a little bit about the nature of data it is returning to you, so the search page, for example, returned a lot of information, the id some key corresponding to what is the, this is, yeah, so in fact, it says this, the key basically says it is the page title in a URL friendly format.

Whereas this title is the page title in a reading friendly format, the primary difference will be that spaces will be replaced by underscores in the key. An excerpt is, for the search pages endpoint, a few lines, it does not specify exactly what it is and a description, which is a short summary of the description, or null if no entry exists. So, in other words, if you searched for something that does not exist, you should get back description equal to null. Not the string null, but the value null.

So, it is actually a Java Script object with the value null. So, this schema, in other words, is something very important. And, this is something we will again look at later. This is part of the documentation of an API.

(Refer Slide Time: 17:05)



The screenshot displays the 'Detailed documentation' for a search API endpoint at IIT Madras. It includes a table with the route `/search/page?q=search terms`, method `GET`, and content type `application/json`. A note specifies that the endpoint uses the search engine configured in the `$wgSearchType` setting. An example section shows a `curl` command for searching Wikipedia for 'Jupiter' with a limit of 20 results. Below this, the 'Parameters and Response Codes' section contains two tables. The first table lists parameters: `q` (required query, search terms) and `limit` (optional query, maximum number of results between 1 and 100, default 50). The second table lists response codes: 200 (Success: Results found), 200 (Success: No results found), 400 (Query parameter not set), 400 (Invalid limit requested), and 500 (Search error).

### Detailed documentation

**Search pages** [edit]

Route	/search/page?q=search terms	Content type	application/json
Method	GET	Returns	pages object containing array of search results

Searches wiki page titles and contents for the provided search terms, and returns matching pages.

**When using this endpoint on your wiki**

This endpoint uses the search engine configured in the `$wgSearchType` configuration setting and returns results in the namespaces configured by `$wgNamespacesToBeSearchedDefault`.

**Examples** [edit]

`curl` Python PHP JavaScript

```
# Search English Wikipedia for up to 20 pages containing information about Jupiter
$ curl https://en.wikipedia.org/w/rest.php/v1/search/page?q=jupiter&limit=20
```

### Parameters and Response Codes

**Parameters** [edit]

q	Search terms
required   query	
limit	Maximum number of search results to return, between 1 and 100. Default: 50
optional   query	

**Responses** [edit]

200	Success: Results found. Returns a <code>pages</code> object containing an array of search results.
200	Success: No results found. Returns a <code>pages</code> object containing an empty array.
400	Query parameter not set. Add <code>q</code> parameter.
400	Invalid limit requested. Set <code>limit</code> parameter to between 1 and 100.
500	Search error

There is some more detailed documentation, which basically says, what is the API for this search page, the route that needs to be used? What does that mean? It basically means that this is the route or the part of the URL that you need to use in order to access this part of the API. What is the method that you should use? GET. What is the content type that you can expect in response?

And what exactly is it that it will return, it will return a `pages` object, which contains an array of search results? It also gives you an example of how to use it. So, I did it with Earth, it says you could also do it with Jupiter and with 20 results and up to 20 pages containing information about Jupiter.

Continuing on the documentation, it also says what are the parameters that can be accepted for this, there is `q` which is required, you have to give it a query string, otherwise, it is basically going to give you some kind of an error. And there is an optional limit, which says that you could return the maximum number of search results to return, what is the default value that it will return? And what is the maximum that it can handle?

So, all of that information is documented cleanly over here. So, someone who wants to use this API, the moment they have seen all this information, that is enough to know exactly what you can do with it. Finally, it also says what are the possible responses that you can get from the server? The ideal status is a response 200 results found, and it returns a `pages` object, which contains an array of search results.

If nothing was found, that is also a successful search. It is not a failure, because what it is saying is that I searched and did not find anything. So, yes, the search succeeded. It is just that there was no data in our in my database. It will still return a `pages` object containing an empty array. But then come the error codes, so the 400 over here is an error code, which will basically come if you do not set a `q` parameter, remember that the `q` was declared as required out here? So, if I do not have a `q` parameter to search for, it will come back with a 400 error.

Similarly, if I give an invalid limit, let us say I asked limit equal to 1000, or limit equal to minus 1, I will get a 400-error saying this is an invalid limit outside the range of what I can accept. And finally, let us say there was an, the so-called internal server error, the server crashed or something happened and the search just failed. It cannot tell you anything better than that. It will give you back a 500 error. So, these are the possible responses that you could get for this API.

So, you see how the documentation works. It basically sort of tells you the route to use the method to use what kind of content type to expect what data to expect in return. The example is a nice touch. It may not always be part of the documentation, but it is always useful if you can give an example, the parameters taken as input essential and the type of responses that you can, response codes that you can expect. All of these are essential information for the documentation.