Day 66 Goals: Build Your Own REST API Service

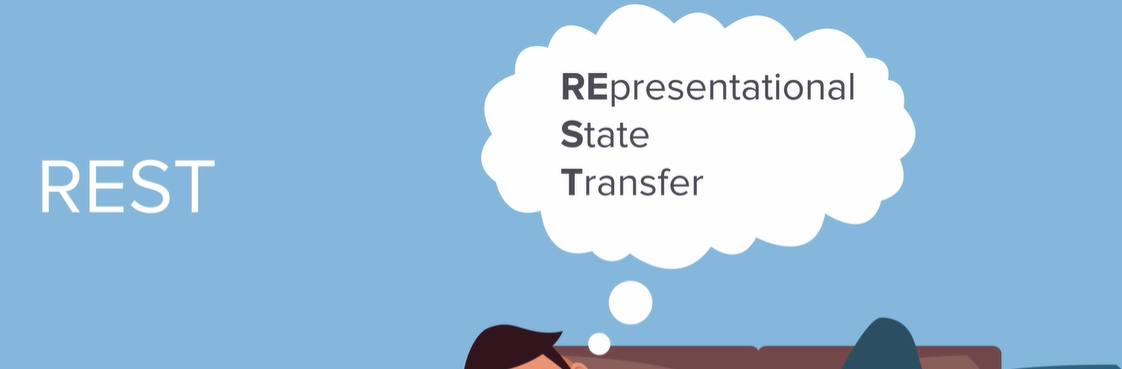
On day 33, we learnt about APIs and since then, we've used a number of public APIs. e.g ISS location, Trivia Questions and Twilio. In a lot of cases, the API allows us to tap into a particular website's data or service.

Many companies have collected valuable data e.g. Bitcoin prices, Restaurant reviews and provide an API for developers to access this data for a price. Depending on how valuable the data/service is behind the API, these APIs can charge anywhere from $9 to $99 per month for access. Some even charge per API call.

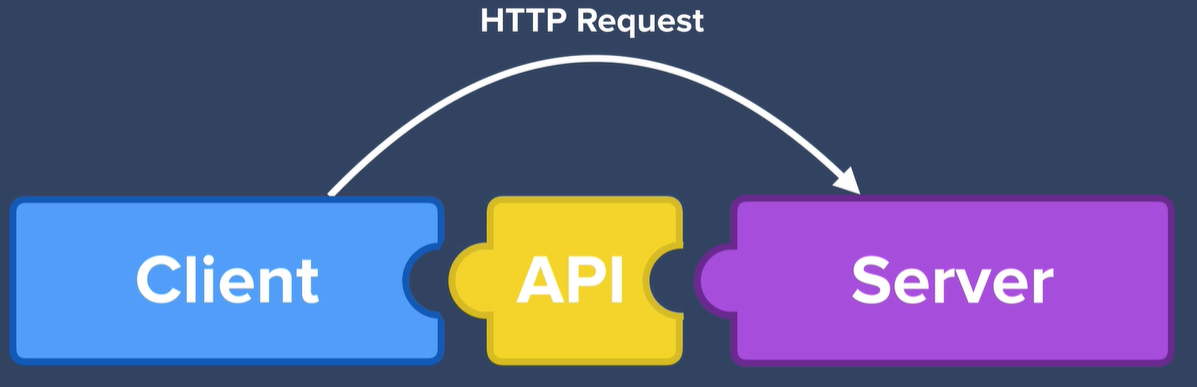
What if you have access to some information that other people might want to use? E.g. You collected data on all the cafes in a particular city and figured out which ones were suitable for remote-work? Then you could create an API and charge people to access your data.

But how do you create an API?

That's what we'll tackle in today's lessons. Building a full REST API from scratch using Flask.



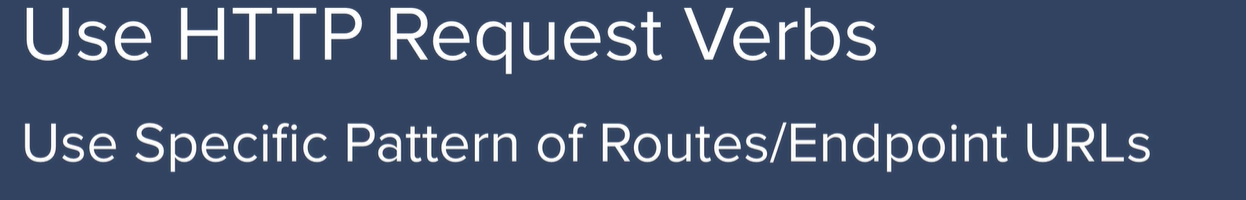


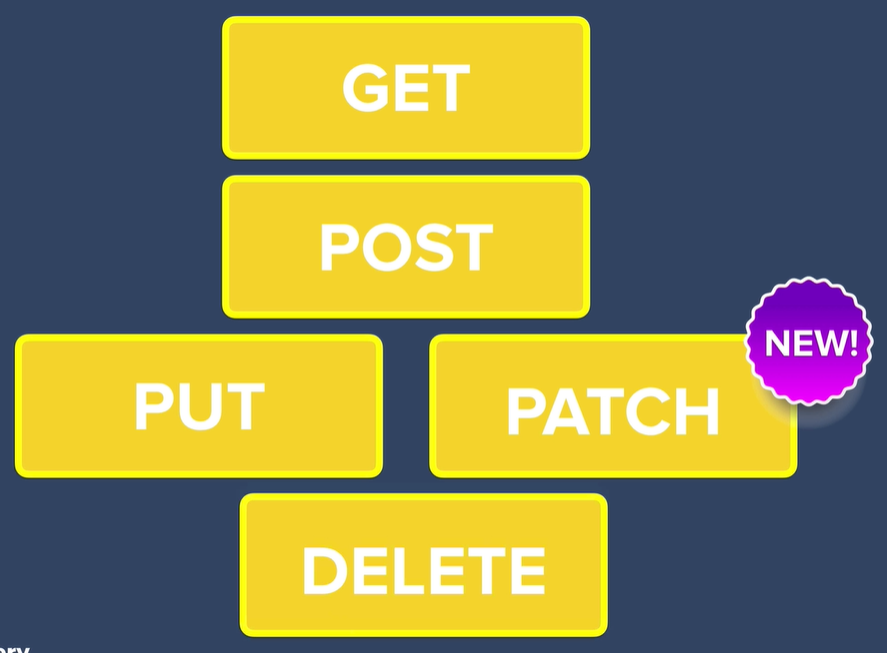
Server will have a bunch of API’s which are services that can be exposed for clients to tap into .

REST is essentially an architectural style for designing APIs,- a set of rule that web developers follow when they are building API’s

it's not the only architectural style. Some others are :

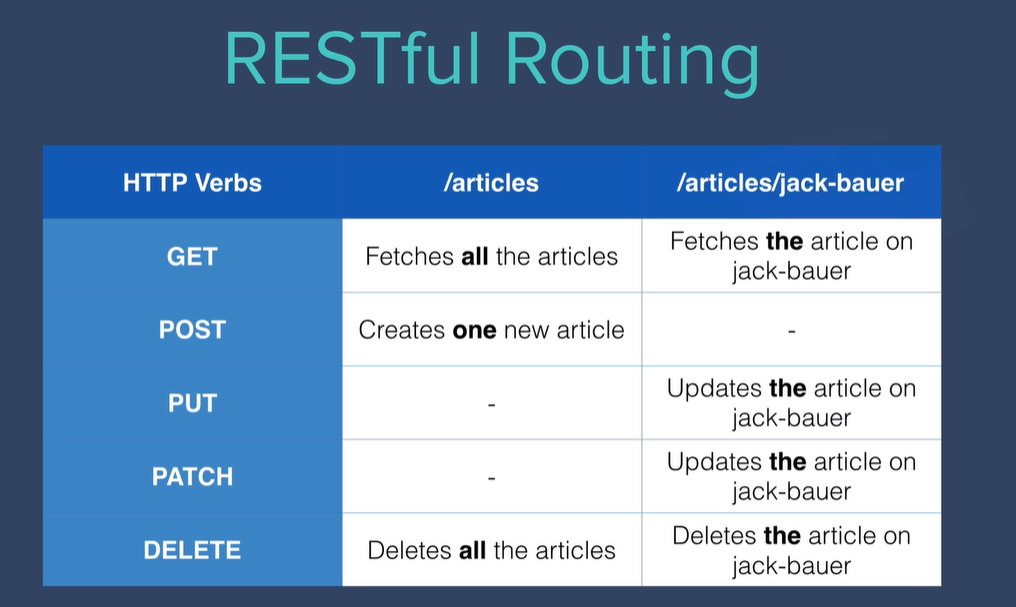


REST full rules – 2 most important parts 

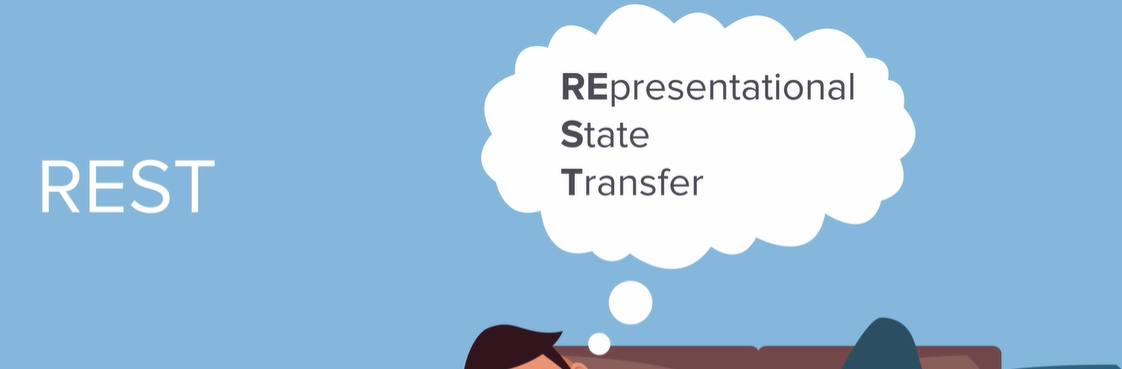


Similar to CRUD





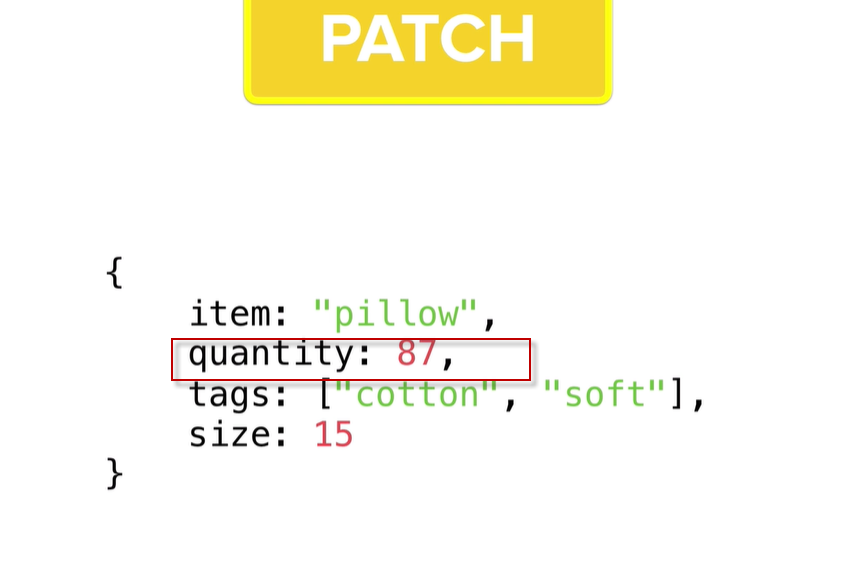
specify specific routes or URL's in order to access certain resources. In order for our API to be RESTful, we have to have a specific pattern of endpoints and routes.



PUT an entirely new



Patch – just the value that needs updating



<https://flask.palletsprojects.com/en/1.1.x/quickstart/#variable-rules>

Build Documentation for Your API

If we want other people to use our API, then we have to document how to use it. People can't see the code on our servers, so we have to tell them how to interact with our servers via the API constraints.

e.g. What are the routes, what are the required parameters etc.

Luckily for us, if you made all your requests in Postman and you gave each request a name and description then Postman will generate the documentation automatically for you.

1. Make sure that you've made each of the requests and they work as you expect.

2. Make sure all the requests are saved in the same collection e.g. My collection is called Cafe & Wifi:

3. Click on the three dots next to your collection name and go to "Publish Docs":

4. Go through the steps to publish your documentation and this is what you should end up with:

e.g. Here's mine: <https://documenter.getpostman.com/view/2568017/TVRhd9qR>

5. We can now edit out index.html to include an anchor tag to our API's documentation.

[COMPLETED PROJECT CODE](https://repl.it/@appbrewery/cafe-api-end#main.py)