November 3, 2021

**APIs**

**Overview**

* What is an API?-a way we can get data
* RESTful API Design-a design pattern how you create these servers
* Accessing APIs
* Postman-acts as a stand in for the front end

**Background**

**API**

* API stands for Application Programming Interface. Interface in this case refers to *how we can interact* with the application, *not a user interface.* So an API is the set of features and rules that allow for interaction. An API depends on how a code interact with it. A userinterface depends on how the person interacts with the code.
* We have already been creating and using APIs throughout this course! The DOM is an API that allows us to interact with web pages. There are APIs for interacting with Bluetooth devices. When we’ve created servers, we were setting up an API and defining how it could be used with endpoints. Those are the kind we’ll be working with today. We’ll learn more about their design and how to interact with them.

**Why APIs?other people servers that are waiting to request data from them.**

* APIs provide tons of information and capabilities to developers that they wouldn’t otherwise have. For example, there are APIs that provide sports data. If you wanted to create a fantasy football website, you could use an API that provided data for you instead of having to go and collect it yourself.
* Or maybe you want to create a custom script for your smart thermometer. By reading through the API documentation, you can figure out what functionality is available to you without knowing how the entire system works.

**How do we use them?**

It depends on the API. Some are accessible right in HTML or JavaScript or through specific software. Today, we’ll be using URLs (endpoints, params, queries, etc) to interact with APIs that are hosted on the internet. We’ll get into more specifics after we talk about API design and REST.

**REST**

**RESTful API-a specific design pattern.**

**Representational State Transfer** Application Programming Interface

It means that a request or response will require/include ALL the data, it doesn’t rely on another request/function

**This sounds important**

It is!

* A set of standards for API architecture
* **URLs are uniform and meaningful- when we send a get resquest for users, we should get data about users.**
* Client and server are independent of each other
  + Can be in different languages!
* HTTP(S) is the protocol
* Each request is independent

**What Does It Look Like?**

RESTful API design looks like the servers we’ve used and created in class so far. So working with other people’s APIs should feel pretty familiar.

**REST Endpoints**

Different methods (usually) produce different results for the same endpoint. Consistent, predictable, and meaningful endpoint naming is an important part of REST.

| **Method** | **Endpoint** | **Usage** |
| --- | --- | --- |
| GET | ***/albums*** | Get a list of albums |
| POST | ***/albums*** | Create a new album |
| GET | ***/albums/<id>*** | Get one album |
| PUT | ***/albums/<id>*** | Edit one album |
| DELETE | ***/albums/<id>*** | Delete one album |

Using a hierarchy in designing endpoints helps us stay organized. Thinking in a hierarchy can help us figure out other people’s APIs.

| **Endpoint** | **Usage** |
| --- | --- |
| ***/albums*** | Get a list of several music albums |
| ***/albums/<id>*** | Get one album by its ***id*** |
| ***/albums/<id>/tracks*** | Get an album’s tracks |

**Accessing Data from APIs**

**How to Get Started**

Look for documentation! Here’s what you can expect:

* List available endpoints
* Tell you how to format request parameters
* Tell you what you’ll get back
* Explain rate limits, authorization requirements, etc

**Example API Docs: Spotify**

[Spotify Web API Documentation](https://developer.spotify.com/documentation/web-api/)

**Know Your Limits!**

Many APIs have rate limits. They can be:

* Per Minute
* Per Hour
* Per Day
* Lifetime

And by:

* IP address
* User ID

**Authentication**

Many APIs require authentication so that they can

* Track requests made
* Generate analytics
* Require permission

**App Authentication an api will allow certain apps to talk to it. Like google maps will only allow google api to talk to it.**

* The application is allowed to speak to the API
* Example: Google Maps

**User Authentication like twitter or facebook**

* The user permits the application to speak to the API on their behalf
* Usually goes hand-in-hand with app authentication
* Often uses OAuth
* Examples: Twitter, Facebook

**How to Keep Going**

As with much of coding, there is lots of trial and error in figuring out how to use APIs. Figuring out documentation can be tough, but you’ll get better at it with time. Keep reading and trying things out.

* Try a few requests to get a feel for the API and the data it returns.
* Think about making the smallest steps possible. Don’t do everything at once.

**Encoding URLs**

Sometimes you’ve got to send special characters or even spaces in a request over the URL.

In a URL, a space is represented as a plus + sign:

I love Python

I+love+Python

Reserved characters are *percent-encoded*.

Soda & Swine

Soda%26Swine

Here are the encodings of some reserved characters:

| **Character** | **Percent-Encoding** |
| --- | --- |
| ! | %21 |
| # | %23 |
| $ | %24 |
| & | %26 |

You can find a full list of reserved characters and their encodings in the [MDN docs on percent-encoding](https://developer.mozilla.org/en-US/docs/Glossary/percent-encoding).

**Postman**

**What is Postman**

Postman is a tool that we can use to interact with APIs. It is kind of a stand in for a front end.

**Layout**

Everything on the top has to do with the request. Everything on the bottom has to do with the response.

*Calendar

Description automatically generated*

In the request section, we can change what type of request we’re making, specify the URL we’re making the request to, and add a body if needed.

*Graphical user interface, text, application, email

Description automatically generated*

In the response section, we can see the data returned to us as well as other details like any cookies or headers.

*Graphical user interface, text, application

Description automatically generated*

**The End**

**POSTMAN INSTRUCTIONS**

In postman just create a new workspace and make sure it’s personal and not team then create workspace

1. First go to website and get api

2. Then copy get request

(if you were attaching a body then you’d click body and json)

For a specific one

<https://pokeapi/co/api/v2/pokemon-color/red/>

base url is included in every api call like pokeapi.co is base URL

to post:

make sure its on raw and json and don’t have to do comma after to execute a new one if you’re adding on to something because you might get an error.

server can connect buyer to the car

/20/buyer

<https://api.fullcontact.com/v3/person.enrich> <<find out your info

in order to run npm init -y