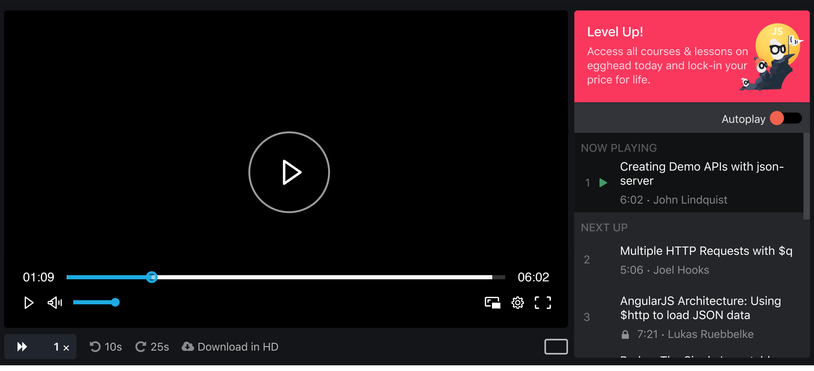

Creating Demo APIs with json-server

SHARE

John Lindquist JavaScript Node.js

json-server makes it extremely easy to setup robust JSON apis to use for demos and proof of concepts. John walks you through the process of using pre-built json files for a server and how to generate larger datasets using lodash and faker.

Transcript Code Discuss

00:01 I'm going to use npm to install globally JSON's server. I could use this anywhere in my system, but I'm just going to use it in this current directory. I'm going to correct a file called DBJSON and describe it as an object with an array of people. It's simply going to be ID zero and name John.

00:31 With this JSON file I can go ahead and type "JSON server" and then "DB.JSON," and this will launch JSON server for me at port 3,000. So if you launch a browser at port 3,000 you can see it gives you a nice little page here saying you can load people. It gives me the array with the single person. Or you can just launch /DB which will give you the entire JSON database.

01:00 It also mention you can get posts put to this JSON file or this JSON data that's loaded in memory, so let's go ahead and try that out. I'm going to use a tool from WebStorm called Test RESTful Web Service. This REST client allows me to put in the URL I want to use which is local host with a port 4,000 and the path I want to use, which is people.

01:27 I'll go ahead and just make a get request of this which will return just what we saw before. If I want to post to people, I'll say, "post, make a new request," which I'm going to set to content type application/JSON with a text of name Henry.

01:53 Then when I hit run you'll see it will post Henry. I can post another one, so it will be number two. Once I get this again and hit run, you can see I get John, Henry, Henry since I ran post twice. Now you can see this last guy has the same name, so let's go ahead and update him.

02:14 We'll go to the path of two here, so slash two. We'll say put, and then in my request we'll change the name to Frank. We'll run this. It looks like I forgot to click this here to set the text to that. Run it again, so now you can see we've got Frank. If I go ahead and I get the entire array of people, you can see now I have John, Henry, and Frank.

02:47 Otherwise if I just wanted to get Frank I could say slash two, hit enter, and get Frank. You can see we already have a fairly robust API to play around with all this data. But we really don't have that much data to work with, so let's go ahead and fake out a whole bunch of data to play around with.

03:04 At this point if I were to shut down my JSON server it would actually lose all of the changes I've made because it's just keeping it in memory. But as it says here, if you hit S...so I'll type S and hit enter. It saves a snapshot that shows all of the changes I made. Then it backs up my previous file named this. We now have DBJSON as this new snapshot.

03:27 We'll go ahead and break this. Now we'll create a JavaScript file which we'll just call generate.JS. This is going require us to export a function which returns some sort of object. Whatever object we get back is going to be used as that JSON data.

03:51 The tools I want to use for this, I'm going to go ahead and say, "npm, install Faker and Lo-Dash." This will help me easily generate this data that I want to generate. I'll go ahead and require Faker, and I'll require Lo-Dash. I'll use these tools, so I'm going to return an object with a people property which needs to be an array, or it's going to be an array that I'm going to return.

04:24 That array is going to be generated by Lo-Dash and Faker. I'm going to use Lo-Dash times 100, meaning run this function 100 times. This argument here is going to be the index as it loops through 100 times. We will return from this an object with an ID of N which matches here.

04:46 The first person is going to be zero. Then we'll return a name of Faker.name.findname. If you Google for Faker or JS, you can find all these different APIs you can use. It will say, "Avatar is Faker.internet.avatar." Now all I have to do is say, "JSON's server" and then target my generate JS script, hit enter.

05:16 You can see it has @port3000, a URL of people. I'll get rid of this ID here. if I open my RESTful Web Services and I run a get request, you can see it returns all of this data, a hundred different people, ID zero, name Lacy, with an avatar to a Twitter avatar. All of this data I could loop through, query, post, update, delete, all that sort of stuff.

05:50 You can even, like I said, query. If I say, "Q equals Randy," hit enter, return, it's going to return everyone named Randy, which was that second guy that I noted. You'll have a ton of data to work with for any demo you want to build.