Web Services (GET): Tutorial (JavaScript)

In this tutorial, you'll work on a document listing application. Most of the functionality to render data is in the starting code. You're responsible for calling the Web API to retrieve a list of documents and details for each one.

Step One: Run the project

The first thing to do is install any dependencies by running the command `npm install`. If you run the project using the command `npm run serve`, the Vue application starts normally on port 8080. The `serve` command also starts up a local Web API on port 3000.

Step Two: Explore the web API

Before moving on to the next step, explore the web API using Postman. You can access the following endpoints:

- GET: http://localhost:3000/docs
  - A list of all documents.
- GET: http://localhost:3000/docs/1
  - Details for a document using the document id.
  - There are 100 documents, each with an id.
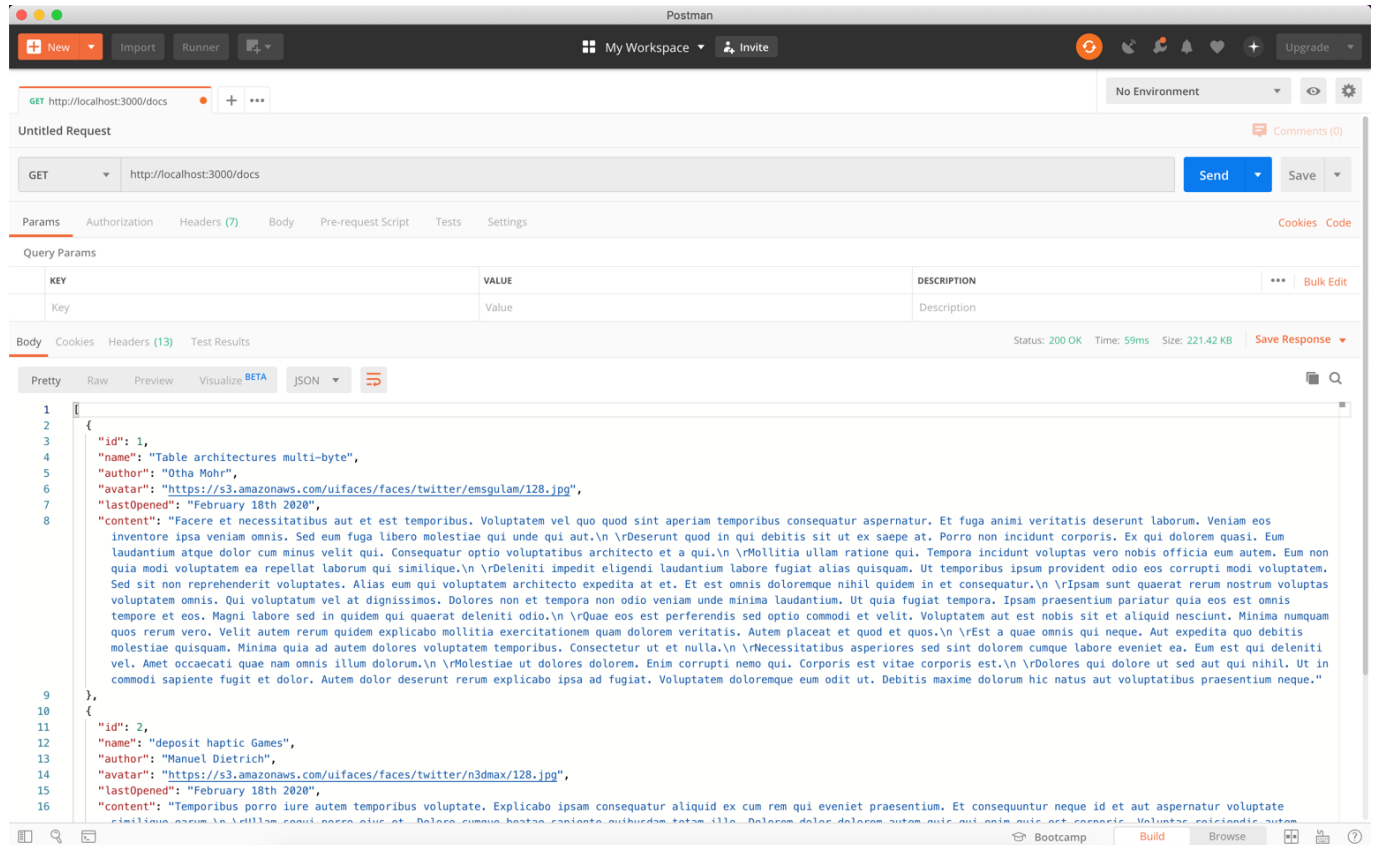
Step Three: Review starting code

Before you get started with the tutorial, take a minute to review the starting code. If you look at
`/src/router/index.js`, there are two routes set up for Home and Document:

```
const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home
  },
  {
    path: '/document/:id',
    name: 'Document',
    component: Document
  }
]
```

The home view `/src/views/Home.vue` imports and uses the `/src/components/DocumentList` component.
This component has all the code necessary to render an array of document objects.

The `docs` array is currently empty, so your first task is to set up a service object and retrieve a list of
documents to populate that array.

Step Four: Create document service object

You need a document service object to retrieve a list of documents. Start by creating a new folder under `src`
called `services` and create a new file called `DocsService.js`. You'll start by importing `axios` and setting the
`baseURL`:

```
import axios from 'axios';

const http = axios.create({
  baseURL: "http://localhost:3000"
});
```

Next, you'll export a default object that has a single method called `list()`. This method uses `axios` to call
the document's Web API and return a `Promise`:

```
import axios from 'axios';

const http = axios.create({
  baseURL: "http://localhost:3000"
});

export default {

  list() {
    return http.get('/docs');
```

```
    }

  }
```

With the `DocsService` file in place, you need to use it inside of the document list component. Open up `/src/components/DocumentList.vue` and import the `DocsService` that you just created:

```javascript
import docsService from "../services/DocsService";

export default {
  name: "document-list",
  data() {
    return {
      docs: []
    };
  },
  methods: {
    viewDocument(id) {
      this.$router.push(`/document/${id}`);
    }
  }
};
```

Next, use the `created()` lifecycle hook to call the `list()` method in `DocsService`. The `list()` method returns a promise, and when it resolves, you can set the `docs` array to the response:

```javascript
import docsService from "../services/DocsService";

export default {
  name: "document-list",
  data() {
    return {
      docs: []
    };
  },
  methods: {
    viewDocument(id) {
      this.$router.push(`/document/${id}`);
    }
  },
  created() {
    docsService.list().then((response) => {
      this.docs = response.data;
    });
  }
};
```

If you visit http://localhost:8080, you should see a list of documents:

My Documents

|	Document Name	Author	Last Opened by me
📄	Table architectures multi-byte	Otha Mohr	February 18th 2020
📄	deposit haptic Games	Manuel Dietrich	February 18th 2020
📄	e-business Borders Dynamic	Nettie Hudson	February 19th 2020
📄	matrices granular reciprocal	Gwen Shanahan	February 18th 2020
📄	Granite Hungary Graphic Interface	Marcelo Stiedemann	February 18th 2020
📄	intangible Shoes secured line	Mylene West	February 18th 2020
📄	zero defect client-server generate	Elroy Koss	February 18th 2020
📄	Cliffs Syrian Pound Small	Kevin O'Kon	February 18th 2020
📄	South Carolina feed Supervisor	Marian Beier	February 19th 2020
📄	Stravenue applications Manager	Misael DuBuque	February 18th 2020
📄	Coordinator bandwidth Sweden	Emiliano Jaskolski	February 19th 2020
📄	Indian Rupee Ngultrum cyan Ball	Rebeca Rolfson	February 19th 2020
📄	User-friendly Total Rustic Metal Chicken	Alvina Gleichner	February 18th 2020
📄	synergistic overriding Awesome	Edward McLaughlin	February 18th 2020

If you hover over a table row, you'll notice that it gets highlighted. If you click on the row, the method `viewDocument()` is called, which uses Vue Router to navigate to the details for that document:

```
<tr v-for="doc in docs" :key="doc.id" v-on:click="viewDocument(doc.id)">
```

```
viewDocument(id) {
  this.$router.push(`/document/${id}`);
}
```

Step Five: Create method to display document details

If you click on the first document from the home view, you are sent to http://localhost:8080/document/1. Right now, the document detail page has no details:
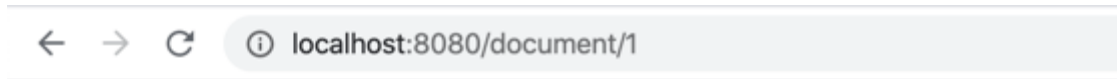
Document Detail

Author: | Last Opened by Me:

[textarea/input area]

Return to Documents List

If you look at the router configuration, there's a route that matches that path. This loads the document view `/src/views/Document`, which uses the `/src/components/DocumentDetail` component:

```
{
  path: '/document/:id',
  name: 'Document',
  component: Document
}
```

The `DocumentDetail` component has the functionality to render the details for a document. To render those details, you need to create a new method in your `DocsService` object and call it from your `DocumentDetail`

component.

First, open up `/src/services/DocsService.js` and add a new method that retrieves a single document by its id:

```javascript
import axios from 'axios';

const http = axios.create({
  baseURL: "http://localhost:3000"
});

export default {

  list() {
    return http.get('/docs');
  },

  get(id) {
    return http.get(`/docs/${id}`)
  }

}
```

Next, open up `/src/component/DocumentDetail.vue` and import the `DocsService`:

```javascript
import docsService from "../services/DocsService";

export default {
  name: "document-detail",
  data() {
    return {
      document: {
        id: null,
        name: "",
        author: "",
        lastOpened: null,
        content: ""
      }
    };
  }
};
```

Finally, use the `created()` method lifecycle hook to call the `get()` method in `DocsService`. Set the component's document property equal to the response of the Web API call:

```javascript
import docsService from "../services/DocsService";

export default {
```

```
    name: "document-detail",
    data() {
      return {
        document: {
          id: null,
          name: "",
          author: "",
          lastOpened: null,
          content: ""
        }
      };
    },
    created() {
      docsService.get(this.$route.params.id).then((response) => {
        this.document = response.data;
      });
    }
  };
```

If you run the application, you should be able to see the list documents and then click on one to view the details for that document:

Document Detail

Table architectures multi-byte

Author: Otha Mohr | **Last Opened by Me:** February 18th 2020

Facere et necessitatibus aut et est temporibus. Voluptatem vel quo quod sint aperiam temporibus consequatur aspernatur. Et fuga animi veritatis deserunt laborum. Veniam eos inventore ipsa veniam omnis. Sed eum fuga libero molestiae qui unde qui aut.

Deserunt quod in qui debitis sit ut ex saepe at. Porro non incidunt corporis. Ex qui dolorem quasi. Eum laudantium atque dolor cum minus velit qui. Consequatur optio voluptatibus architecto et a qui.

Mollitia ullam ratione qui. Tempora incidunt voluptas vero nobis officia eum autem. Eum non quia modi voluptatem ea repellat laborum qui similique.

Deleniti impedit eligendi laudantium labore fugiat alias quisquam. Ut temporibus ipsum provident odio eos corrupti modi voluptatem. Sed sit non reprehenderit voluptates. Alias eum qui voluptatem architecto expedita at et. Et est omnis doloremque nihil quidem in et consequatur.

Ipsam sunt quaerat rerum nostrum voluptas voluptatem omnis. Qui voluptatum vel at dignissimos. Dolores non et tempora non odio veniam unde minima laudantium. Ut quia fugiat tempora. Ipsam praesentium pariatur quia eos est omnis tempore et eos. Magni labore sed in quidem qui quaerat deleniti odio.

Quae eos est perferendis sed optio commodi et velit. Voluptatem aut est nobis sit et aliquid nesciunt. Minima numquam quos rerum vero. Velit autem rerum quidem explicabo mollitia exercitationem quam dolorem veritatis. Autem placeat et quod et quos.

Est a quae omnis qui neque. Aut expedita quo debitis molestiae quisquam. Minima quia ad autem dolores voluptatem temporibus. Consectetur ut et nulla.

Necessitatibus asperiores sed sint dolorem cumque labore eveniet ea. Eum est qui deleniti vel. Amet occaecati quae nam omnis illum dolorum.

Molestiae ut dolores dolorem. Enim corrupti nemo qui. Corporis est vitae corporis est.

Dolores qui dolore ut sed aut qui nihil. Ut in commodi sapiente fugit et dolor. Autem dolor deserunt rerum explicabo ipsa ad fugiat. Voluptatem doloremque eum odit ut. Debitis maxime dolorum hic natus aut voluptatibus praesentium neque.

Return to Documents List

Congratulations, you just hooked up your first Web API to a Vue application.

Summary

In this tutorial, you learned how to:

- Make an HTTP GET request to a Web API using the Axios library.
- Handle the response (Promise) of a Web API request.
- Build a service object for interacting with a Web API.
- Use the `created()` lifecycle hook to call a Web API and retrieve data when the view is rendered.