To work with useEffect and axios in React, here's how to handle both GET and POST requests with cleanup logic. I’ll also create a dummy API example for both.

Setting Up

Ensure you have axios installed in your project:

bash

Copy code

npm install axios

Then, in your React component, you can use useEffect with axios to fetch or post data. Here’s a basic setup:

javascript

Copy code

import React, { useEffect, useState }

You said:

use react hook forms

ChatGPT said:

ChatGPT

Using react-hook-form with axios in React is efficient for handling forms, especially when working with POST requests. I'll guide you through a simple setup to handle both GET and POST requests with cleanup logic using useEffect.

Setting Up

First, make sure you have react-hook-form and axios installed:

bash

Copy code

npm install react-hook-form axios

1. Basic Setup for the Component

Here's an example where we:

Fetch data (GET request) from a dummy API when the component mounts.

Use react-hook-form to handle form submission (POST request).

Clean up **using** useEffect.

Code Example

javascript

Copy code

import React, { useEffect, useState } from 'react';

import axios from 'axios';

import { useForm } from 'react-hook-form';

**const** YourComponent = () **=>** {

**const** [data, setData] = useState([]);

**const** { register, handleSubmit, reset } = useForm();

**const** [loading, setLoading] = useState(true);

**const** [error, setError] = useState(null);

*// 1. GET Request using useEffect*

  useEffect(() **=>** {

**const** source = axios.CancelToken.source(); *// Create a cancel token for cleanup*

**const** fetchData = **async** () **=>** {

      setLoading(true);

      try {

**const** response = await axios.get('https://jsonplaceholder.typicode.com/posts', {

          cancelToken: source.token, *// Attach the cancel token*

        });

        setData(response.data);

      } catch (err) {

        if (axios.isCancel(err)) {

          console.log('Request canceled', err.message);

        } else {

          setError('Failed to fetch data');

        }

      } finally {

        setLoading(false);

      }

    };

    fetchData();

    return () **=>** {

      source.cancel('Component unmounted, request canceled');

    };

  }, []);

*// 2. POST Request using react-hook-form*

**const** onSubmit = **async** (formData) **=>** {

    try {

**const** response = await axios.post('https://jsonplaceholder.typicode.com/posts', formData);

      console.log('Posted data:', response.data);

      setData((prevData) **=>** [response.data, ...prevData]); *// Add the new data to the top of the list*

      reset(); *// Clear the form after submission*

    } catch (err) {

      console.error('Error posting data:', err);

      setError('Failed to submit data');

    }

  };

  return (

    <div>

      <h2>Fetched Data</h2>

      {loading && <p>Loading...</p>}

      {error && <p>{error}</p>}

      <ul>

        {data.map((item) **=>** (

          <li key={item.id}>{item.title}</li>

        ))}

      </ul>

      <h2>Submit Data</h2>

      <form onSubmit={handleSubmit(onSubmit)}>

        <label>

          Title:

          <input {...register('title', { required: true })} />

        </label>

        <label>

          Body:

          <textarea {...register('body', { required: true })} />

        </label>

        <button type="submit">Submit</button>

      </form>

    </div>

  );

};

export default YourComponent;

Explanation

GET Request with Cleanup:

The useEffect hook makes a GET request to fetch data when the component mounts.

We create an axios cancel token and pass it to the request, allowing us to cancel it when the component unmounts.

POST Request with react-hook-form:

We use react-hook-form to handle the form state and submission.

The onSubmit **function** is called when the form is submitted. This function sends a POST request using the entered data.

After a successful submission, we reset the form and update the data state with the new entry.

Error Handling:

Simple error handling is implemented for both the GET and POST requests.

You said:

<h2>Submit Data</h2>

      <form onSubmit={handleSubmit(onSubmit)}> i didnt understand this line where is data sent

ChatGPT said:

ChatGPT

The line you're asking about is part of the form submission process, and it's controlled by react-hook-form. Let’s break it down to clarify where and how the data is sent.

Explanation of handleSubmit and onSubmit

handleSubmit(onSubmit):

handleSubmit is a method provided by react-hook-form. It wraps the form’s submit event and takes care of validating the form fields before calling your custom function—in this case, onSubmit.

When the form is submitted, handleSubmit gathers all the form data, validates it according to the rules set in the register function (like { required: true }), and then calls the onSubmit function if validation passes.

onSubmit Function:

This is the custom function where we specify what to do with the form data after submission.

In this example, onSubmit is an asynchronous function that uses axios.post to send the form data to an external API (a POST request to 'https://jsonplaceholder.typicode.com/posts').

Here's that part of the code again for reference:

javascript

Copy code

// onSubmit is the function that handles the form data after submission

const onSubmit = async (formData) => {

  try {

*// Send a POST request with formData*

    const response = await axios.post('https://jsonplaceholder.typicode.com/posts', formData);

    console.log('Posted data:', response.data);

    setData((prevData) **=>** [response.data, ...prevData]); *// Add the new data to the list*

    reset(); *// Clear the form after submission*

  } catch (err) {

    console.error('Error posting data:', err);

    setError('Failed to submit data');

  }

};

What Happens When You Submit the Form

When the user clicks Submit:

handleSubmit collects all form data into an object (e.g., { title: 'some title', body: 'some body text' }).

It calls the onSubmit function with this data.

Inside onSubmit:

axios.post sends the collected formData to 'https://jsonplaceholder.typicode.com/posts'.

axios sends this as the POST request's payload (or request body).

If the request is successful, we log the response and update the data state to include the new item, displaying it in the list.

reset() clears the form fields.

This flow allows you to take form data entered by the user and send it to an API when the form is submitted.

You said:

my question is how is onsubmi function got data

ChatGPT said:

ChatGPT

Got it! Let’s dive deeper into how the onSubmit function receives the form data.

How onSubmit Gets Data from handleSubmit

The form data is collected and provided to the onSubmit function through handleSubmit in react-hook-form.

Here’s the detailed process:

Form Data Collection by handleSubmit:

When you wrap the onSubmit function in handleSubmit, it tells react-hook-form to collect all the data from the form fields registered via register.

react-hook-form automatically gathers values from each registered input and compiles them into an object, which we call formData.

Passing Data to onSubmit:

After collecting the form data, handleSubmit will call onSubmit and pass the collected data to it as an argument.

In this setup, formData is the parameter of onSubmit (i.e., onSubmit = async (formData) => { ... }), and it receives the data object automatically from handleSubmit.

For example:

javascript

Copy code

// If form data entered is { title: "New Post", body: "This is a post body." }

const onSubmit = async (formData) => {

    console.log(formData);

*// Output: { title: "New Post", body: "This is a post body." }*

};

Putting It All Together in the Code

In this line:

javascript

Copy code

<form onSubmit={handleSubmit(onSubmit)}>

handleSubmit(onSubmit):

Runs when the form is submitted.

Collects the data from registered fields (title and body in this case).

Passes this collected data object to onSubmit as the formData parameter.

Here’s a Simplified Flow

User fills in the form fields (e.g., title and body).

When the form is submitted, handleSubmit:

Collects { title: value, body: value } from the form.

Passes this object as formData to onSubmit.

onSubmit now has the form data in the formData parameter and can use it (e.g., to send it via axios).

Example to Illustrate This

Here’s a short example where you can log the form data:

javascript

Copy code

const onSubmit = async (formData) => {

  console.log(formData); *// Logs collected data from the form fields*

*// Expected output: { title: "user input title", body: "user input body" }*

};

By using handleSubmit(onSubmit), the onSubmit function is triggered with all form data automatically provided as its parameter. This approach simplifies handling form data in React forms.

You said:

onSubmit but here you are just calling fuction but not giving parameter

ChatGPT said:

ChatGPT

You're absolutely right to notice this subtle detail. In the line:

javascript

Copy code

<form onSubmit={handleSubmit(onSubmit)}>

It might look like we're not passing any parameters to onSubmit, but here’s the key: handleSubmit automatically collects and passes the form data to onSubmit as an argument.

How This Works in Detail

handleSubmit(onSubmit) is a higher-order function. This means that:

handleSubmit wraps onSubmit in a way that, when the form is submitted, handleSubmit:

Gathers all the data from the form fields that are registered (in this case, fields registered with register).

Validates the fields based on any rules you set up (e.g., { required: true }).

Calls onSubmit with the collected data as its argument.

So while it looks like onSubmit is being called without any arguments, it’s actually being called with the form data as a parameter under the hood.

Example to Illustrate It

Here’s a simplified example to show what’s happening under the hood:

javascript

Copy code

// Define the onSubmit function to accept formData

const onSubmit = (formData) => {

  console.log(formData); *// This will log the collected form data*

};

// In this line, `handleSubmit` is invoked

// It will gather the form data and then call `onSubmit(formData)` internally

<form onSubmit={handleSubmit(onSubmit)}>

When the form is submitted:

handleSubmit gathers all the form data and validates it.

After validation, it will effectively call:

javascript

Copy code

onSubmit({ title: "User input title", body: "User input body" });

This way, you don’t have to pass formData explicitly. handleSubmit takes care of it for you, keeping the code clean and concise.

Summary

handleSubmit(onSubmit) looks like it doesn’t provide any data, but it actually does.

handleSubmit automatically collects, validates, and passes the form data to onSubmit as its argument.

So onSubmit receives formData without you having to explicitly pass it.