Create (Part II)

Let's tie in the creation to our **React front-end**. We can change our `PersonForm.js` to have a form, now.

This component will be the form that we can add a person to and make a request to our API.

PersonForm Component

Code Block 1 - client/src/components/PersonForm.js

```
import React, { useState } from 'react'
import axios from 'axios';
const PersonForm = () => {
    //keep track of what is being typed via useState hook
    const [firstName, setFirstName] = useState("");
    const [lastName, setLastName] = useState("");
    //handler when the form is submitted
    const onSubmitHandler = (e) => {
        //prevent default behavior of the submit
        e.preventDefault();
        //make a post request to create a new person
        axios.post('http://localhost:8000/api/people', {
            firstName,    // this is shortcut syntax for firstName: firstName,
            lastName      // this is shortcut syntax for lastName: lastName
        })
            .then(res=>{
                console.log(res); // always console log to get used to tracking your data!
                console.log(res.data);
            })
            .catch(err=>console.log(err))
    }

    return (
        <form onSubmit={onSubmitHandler}>
            <p>
                <label>First Name</label><br/>
                {/* When the user types in this input, our onChange synthetic event
                    runs this arrow function, setting that event's target's (input)
                    value (what's typed into the input) to our updated state   */}
                <input type="text" onChange = {(e)=>setFirstName(e.target.value)}/>
            </p>
            <p>
                <label>Last Name</label><br/>
                <input type="text" onChange = {(e)=>setLastName(e.target.value)}/>
            </p>
            <input type="submit"/>
        </form>
    )
}
export default PersonForm;
```

Now, we have a functional (albeit not a very interesting looking) form.

Now you can test out your form to see if it is working. You should be able to see the response in your console. This is why it is so important to use console logs as you work. Learning to watch the movement of information is an integral part of being a software developer!

Testing through Postman and our browser we can see that we are adding new documents to the database, but we want them to display immediately on our React front-end. We will see that process in our next module!

Previous

Next

Privacy Policy