Steps to implement in– SPRING BOOT REST API

The previous example (Revision of SPRING BOOT – REST API)

@CrossOrigin(origins = "http://localhost:3000")

@RestController

@RequestMapping("/users")

public class UsersController {

@Autowired

private UserRepository usersRepository;

@GetMapping("/all")

public List<Users> getAll()

{

return usersRepository.findAll();

}

@GetMapping("/{name}")

public Users getByName(@PathVariable final String name)

{

return usersRepository.getByName(name);

}

@PostMapping("/")

public Users save(@RequestBody Users users)

{

usersRepository.save(users);

return usersRepository.getByName(users.getName());

}

}

Hit this "**http://localhost:8080/users/all** " link in a browser will populate list of users as JSON:

# Build React JS Frontend Application

## React Service Component - REST API Call

For our API calls, here used is  Axios. Below is the npm command to install Axios.

npm add axios

Below is the *UserService.js* service implementation to make our HTTP REST call via Axios.

Our backend User endpoint is available at [**http://localhost:8080/users/all**](http://localhost:8080/users/all).

import axios from 'axios'

const USERS\_REST\_API\_URL = 'http://localhost:8080/api/users';

class UserService {

getUsers(){

return axios.get(USERS\_REST\_API\_URL);

}

}

export default new UserService();

Let's create a *UserComponent.js* file and add the following code to it.

import React from 'react';

import UserService from '../services/UserService';

class UserComponent extends React.Component {

constructor(props){

super(props)

this.state = {

users:[]

}

}

**componentDidMount(){**

**UserService.getUsers()**

**.then((response) => {**

**this.setState({ users: response.data})**

**});**

**}**

render(){

return(

<div>

…

<tbody>

{

this.state.users.map(

user =>

<tr key = {user.id}>

<td> {user.id}</td>

<td> {user.firstName}</td>

<td> {user.lastName}</td>

<td> {user.email}</td>

</tr>

)

}

</tbody>

</div>

)

}

* *constructor()* - The *constructor ()* is invoked before the component is mounted. In the constructor, we have declared our state variables and bind the different methods so that they are accessible from the state inside of the render() method.
* *componentDidMount()* - The *componentDidMount()* is called as soon as the component is mounted and ready.
* *render()* - The *render()* method is the most used lifecycle method. The *render()* method that actually outputs HTML to the DOM.
* We are using the **map operator** to loop over our user list and create the view like:

{

this.state.users.map(

user =>

<tr key = {user.id}>

<td> {user.id}</td>

<td> {user.firstName}</td>

<td> {user.lastName}</td>

<td> {user.email}</td>

</tr>

)

}

For reference :- <https://www.javaguides.net/2020/07/react-js-spring-boot-rest-api-example-tutorial.html>

Pokeman Rest Api calls

Pokeapi.co

Npm install axios

ComA.jsx



