**How to create a service in java that can be used in react**

To create a service in Java that can be used in a React application, you typically create a RESTful API using a framework like Spring Boot in Java.

The React app can then interact with this service by making HTTP requests (GET, POST, PUT, DELETE, etc.).

Here's a step-by-step guide:

**1. Set Up the Java Service with Spring Boot**

**a. Create a Spring Boot Project**

* Use Spring Initializr (<https://start.spring.io/>) to generate a new Spring Boot project.
* Include dependencies like Spring Web, Spring Data JPA (if you need a database), and H2 or MySQL (depending on your database).

**b. Create a Model Class**

This is a simple Java class representing an entity.

package com.example.demo.model;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

**package** com.spring.demo.model;

**import** jakarta.persistence.Column;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.GeneratedValue;

**import** jakarta.persistence.GenerationType;

**import** jakarta.persistence.Id;

**import** jakarta.persistence.Table;

@Entity

@Table(name="products")

**public** **class** Product {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name="product\_id")

**private** Long id;

@Column(name="product\_name")

**private** String name;

@Column(name="price")

**private** Double price;

// Getters and Setters

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Double getPrice() {

**return** price;

}

**public** **void** setPrice(Double price) {

**this**.price = price;

}

}

**c. Create a Repository Interface**

This interface will handle the CRUD operations.

package com.example.demo.repository;

import com.example.demo.model.Product;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<Product, Long> {

}

**d. Create a Service Class**

The service class will contain the business logic.

package com.example.demo.service;

import com.example.demo.model.Product;

import com.example.demo.repository.ProductRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class ProductService {

@Autowired

private ProductRepository productRepository;

public List<Product> getAllProducts() {

return productRepository.findAll();

}

public Product getProductById(Long id) {

return productRepository.findById(id).orElse(null);

}

public Product saveProduct(Product product) {

return productRepository.save(product);

}

public void deleteProduct(Long id) {

productRepository.deleteById(id);

}

}

**e. Create a Controller Class**

This class will handle incoming HTTP requests.

package com.example.demo.controller;

import com.example.demo.model.Product;

import com.example.demo.service.ProductService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

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

@RestController

@RequestMapping("/api/products")

public class ProductController {

@Autowired

private ProductService productService;

@GetMapping

public List<Product> getAllProducts() {

return productService.getAllProducts();

}

@GetMapping("/{id}")

public Product getProductById(@PathVariable Long id) {

return productService.getProductById(id);

}

@PostMapping

public Product createProduct(@RequestBody Product product) {

return productService.saveProduct(product);

}

@PutMapping("/{id}")

public Product updateProduct(@PathVariable Long id, @RequestBody Product product) {

product.setId(id);

return productService.saveProduct(product);

}

@DeleteMapping("/{id}")

public void deleteProduct(@PathVariable Long id) {

productService.deleteProduct(id);

}

}

**f. application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/products\_nagarjuna\_db

spring.datasource.username=root

spring.datasource.password=mysql

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

**2. Run the Spring Boot Application**

* Run the Spring Boot application. By default, it will run on http://localhost:8080.

**3. Connect React with the Java Service**

**npx create-react-app nag-products-app**

**npm install axios**

**a. Make HTTP Requests in React**

You can use **fetch or axios** to interact with the Java service.

Here’s an example using axios:

import axios from 'axios';

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

function App() {

const [products, setProducts] = useState([]);

useEffect(() => {

axios.get('http://localhost:8080/api/products')

.then(response => setProducts(response.data))

.catch(error => console.error('Error fetching products:', error));

}, []);

return (

<div>

<h1>Products</h1>

<ul>

{products.map(product => (

<li key={product.id}>

{product.name} - ${product.price}

</li>

))}

</ul>

</div>

);

}

export default App;

**4. Deploy and Test**

* Test your React application by running npm start and ensuring it can communicate with the Java service.
* Deploy the Java service and React application to a production environment (like Heroku, AWS, etc.).

**Summary**

You’ve set up a Spring Boot service in Java that provides a RESTful API, and you’ve connected a React frontend to this service by making HTTP requests. This architecture allows the React application to interact with the backend service to perform CRUD operations.

**Now develop a full application for all the CRUD operations**

**npm install axios**

Here's how to build the React CRUD app , we'll use custom CSS files for styling each component.

Below is a complete example with:

**Product CRUD operations**

**Custom .css files for each component**

**A clean and modern layout**

**Backend Assumption**

You already have a Spring Boot REST API with the following endpoints:

Method Endpoint Description

GET /api/products Get all products

GET /api/products/{id} Get product by ID

POST /api/products Add product

PUT /api/products/{id} Update product

DELETE /api/products/{id} Delete product

**Directory Structure**

src/

├── App.js

├── services/

│ └── ProductService.js

└── components/

├── ProductList.js

├── ProductForm.js

├── ProductList.css

└── ProductForm.css

**1. ProductService.js**

// src/services/ProductService.js

import axios from 'axios';

const BASE\_URL = 'http://localhost:8080/api/products';

class ProductService {

getAll() {

return axios.get(BASE\_URL);

}

getById(id) {

return axios.get(`${BASE\_URL}/${id}`);

}

create(product) {

return axios.post(BASE\_URL, product);

}

update(id, product) {

return axios.put(`${BASE\_URL}/${id}`, product);

}

delete(id) {

return axios.delete(`${BASE\_URL}/${id}`);

}

}

export default new ProductService();

**2. ProductList.js**

// src/components/ProductList.js

import React from 'react';

import './ProductList.css';

const ProductList = ({ products, onEdit, onDelete, onAdd }) => {

return (

<div className="product-list">

<h1>Product List</h1>

<button className="add-btn" onClick={onAdd}>+ Add Product</button>

<div className="table">

<div className="table-header">

<span>ID</span>

<span>Name</span>

<span>Price</span>

<span>Actions</span>

</div>

{products.map(product => (

<div className="table-row" key={product.id}>

<span>{product.id}</span>

<span>{product.name}</span>

<span>${product.price}</span>

<span>

<button className="edit-btn" onClick={() => onEdit(product)}>Edit</button>

<button className="delete-btn" onClick={() => onDelete(product.id)}>Delete</button>

</span>

</div>

))}

</div>

</div>

);

};

export default ProductList;

**3. ProductForm.js**

// src/components/ProductForm.js

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

import './ProductForm.css';

const ProductForm = ({ productToEdit, onSave, onCancel }) => {

const [product, setProduct] = useState({ name: '', price: '' });

useEffect(() => {

if (productToEdit) {

setProduct(productToEdit);

} else {

setProduct({ name: '', price: '' });

}

}, [productToEdit]);

const handleChange = (e) => {

const { name, value } = e.target;

setProduct(prev => ({ ...prev, [name]: value }));

};

const handleSubmit = (e) => {

e.preventDefault();

onSave(product);

};

return (

<div className="product-form">

<h2>{productToEdit ? 'Edit Product' : 'Add Product'}</h2>

<form onSubmit={handleSubmit}>

<label>Name:</label>

<input

type="text"

name="name"

value={product.name}

onChange={handleChange}

required

/>

<label>Price:</label>

<input

type="number"

name="price"

value={product.price}

onChange={handleChange}

step="0.01"

required

/>

<button type="submit">{productToEdit ? 'Update' : 'Create'}</button>

<button type="button" onClick={onCancel}>Cancel</button>

</form>

</div>

);

};

export default ProductForm;

**4. App.js**

// src/App.js

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

import ProductService from './services/ProductService';

import ProductList from './components/ProductList';

import ProductForm from './components/ProductForm';

const App = () => {

const [products, setProducts] = useState([]);

const [view, setView] = useState('list'); // 'list' or 'form'

const [editingProduct, setEditingProduct] = useState(null);

useEffect(() => {

ProductService.getAll().then(res => setProducts(res.data));

}, []);

const handleAdd = () => {

setEditingProduct(null);

setView('form');

};

const handleEdit = (product) => {

setEditingProduct(product);

setView('form');

};

const handleDelete = (id) => {

ProductService.delete(id).then(() => {

setProducts(prev => prev.filter(p => p.id !== id));

});

};

const handleSave = (product) => {

if (product.id) {

ProductService.update(product.id, product).then(() => {

setProducts(prev => prev.map(p => (p.id === product.id ? product : p)));

setView('list');

});

} else {

ProductService.create(product).then((res) => {

setProducts(prev => [...prev, res.data]);

setView('list');

});

}

};

const handleCancel = () => {

setView('list');

};

return (

<div>

{view === 'list' ? (

<ProductList

products={products}

onEdit={handleEdit}

onDelete={handleDelete}

onAdd={handleAdd}

/>

) : (

<ProductForm

productToEdit={editingProduct}

onSave={handleSave}

onCancel={handleCancel}

/>

)}

</div>

);

};

export default App;

**5. ProductList.css**

.product-list {

width: 90%;

max-width: 800px;

margin: 30px auto;

font-family: sans-serif;

}

.product-list h1 {

text-align: center;

color: #444;

}

.add-btn {

display: inline-block;

padding: 10px 15px;

background-color: #28a745;

color: white;

margin-bottom: 15px;

border: none;

border-radius: 5px;

cursor: pointer;

}

.table {

display: flex;

flex-direction: column;

}

.table-header, .table-row {

display: grid;

grid-template-columns: 1fr 3fr 2fr 3fr;

padding: 10px;

border-bottom: 1px solid #ccc;

}

.table-header {

background-color: #f0f0f0;

font-weight: bold;

}

.edit-btn, .delete-btn {

margin-right: 10px;

padding: 5px 10px;

border: none;

border-radius: 4px;

cursor: pointer;

}

.edit-btn {

background-color: #007bff;

color: white;

}

.delete-btn {

background-color: #dc3545;

color: white;

}

**6. ProductForm.css**

.product-form {

width: 90%;

max-width: 500px;

margin: 30px auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 8px;

background-color: #fafafa;

font-family: sans-serif;

}

.product-form h2 {

text-align: center;

margin-bottom: 20px;

color: #333;

}

.product-form form {

display: flex;

flex-direction: column;

}

.product-form label {

margin-bottom: 5px;

font-weight: bold;

color: #555;

}

.product-form input {

padding: 10px;

margin-bottom: 15px;

border: 1px solid #ccc;

border-radius: 4px;

}

.product-form button {

padding: 10px;

margin-right: 10px;

background-color: #28a745;

border: none;

color: white;

font-weight: bold;

border-radius: 5px;

cursor: pointer;

}

.product-form button[type="button"] {

background-color: #6c757d;

}

**Step-by-Step Commands to Set Up the Project**

**1. Create a New React App**

npx create-react-app product-crud-app

cd product-crud-app

**2. Install Axios for HTTP Requests**

npm install axios

**3. Create Folder Structure**

Create a folder in VSCode as shown below

src/components src/services

**4. Add JavaScript Files**

Create the following files inside the respective folders:

* src/services/ProductService.js
* src/components/ProductList.js
* src/components/ProductForm.js
* Replace the contents of src/App.js with the full App code.

You can create these using a code editor (like VS Code), or via terminal:

src/services/ProductService.js

src/components/ProductList.js

src/components/ProductForm.js

Then, paste the code you have for each of those files.

**5. Optional: Add CSS Files**

If you are using CSS:

src/components/ProductList.css

src/components/ProductForm.css

And include styles accordingly. Don't forget to import them in the components.

**6. Start the React App**

npm start

This will run the app in development mode and open it in your browser at http://localhost:3000.

**Backend API Reminder**

Make sure you have a backend running at:

http://localhost:8080/api/products

This should support:

* GET /api/products
* POST /api/products
* GET /api/products/{id}
* PUT /api/products/{id}
* DELETE /api/products/{id}