Requests left today: 0

MENU

Close menu

Night mode

Select language

Hello! How can I help you?

accepting and rejecting using angular and spring boot

To implement accept/reject functionality using Angular and Spring Boot, you could follow these steps:

First, create a new Angular project using the Angular CLI (ng new <project-name> ).

Create a new service (ng generate service <service-name> ) to communicate with the backend API.

In the service, create two methods to handle the accept/reject operations. These methods should make HTTP requests to the backend API using Angular's HttpClient .

acceptRequest(requestId: number): Observable<any> {

return this.http.patch<any>(`${API\_URL}/requests/${requestId}/accept`, {});

}

rejectRequest(requestId: number): Observable<any> {

return this.http.patch<any>(`${API\_URL}/requests/${requestId}/reject`, {});

}

In your Angular component, inject the service and use it to handle the accept/reject buttons.

import { Component } from '@angular/core';

import { ApiService } from './api.service';

@Component({

selector: 'app-component',

template: `

<div \*ngFor="let request of requests">

{{ request.title }}

<button (click)="acceptRequest(request.id)">Accept</button>

<button (click)="rejectRequest(request.id)">Reject</button>

</div>

`,

})

export class AppComponent {

requests = [];

constructor(private apiService: ApiService) {}

acceptRequest(requestId: number) {

this.apiService.acceptRequest(requestId).subscribe((response) => {

console.log('Request accepted', response);

// Update the requests list with the new request status

});

}

rejectRequest(requestId: number) {

this.apiService.rejectRequest(requestId).subscribe((response) => {

console.log('Request rejected', response);

// Update the requests list with the new request status

});

}

}

On the backend side (Spring Boot), create two REST endpoints to handle the accept/reject requests. You can use Spring MVC annotations to define the endpoints and the request type.

@RestController

@RequestMapping("/requests")

public class RequestController {

@PatchMapping("/{requestId}/accept")

public void acceptRequest(@PathVariable Long requestId) {

// Accept the request with the given ID

}

@PatchMapping("/{requestId}/reject")

public void rejectRequest(@PathVariable Long requestId) {

// Reject the request with the given ID

}

}

Finally, implement the accept/reject logic in the backend service. You can use Spring Data JPA to access the database and update the Request entity status accordingly.

@Service

public class RequestService {

@Autowired

private RequestRepository requestRepository;

public void acceptRequest(Long requestId) {

Request request = requestRepository.findById(requestId).orElseThrow(RequestNotFoundException::new);

request.setStatus(RequestStatus.ACCEPTED);

requestRepository.save(request);

}

public void rejectRequest(Long requestId) {

Request request = requestRepository.findById(requestId).orElseThrow(RequestNotFoundException::new);

request.setStatus(RequestStatus.REJECTED);

requestRepository.save(request);

}

}

That's it! With these steps, you have implemented accept/reject functionality using Angular and Spring Boot.

**SENDING EMAIL USING SPRING BOOT AND ANGULAR**

To implement email sending functionality using Spring Boot and Angular, you can follow these steps:

First, create a new Angular project using the Angular CLI (ng new <project-name> ).

Create a new service (ng generate service <service-name> ) to communicate with the backend API.

In the service, create a method to send the email. This method should make an HTTP request to the backend API using Angular's HttpClient . You could use a POST request to send email data as a JSON payload.

sendEmail(formData: any): Observable<any> {

return this.http.post<any>(`${API\_URL}/send-email`, formData);

}

In your Angular component, inject the service and use it to handle the form submission.

import { Component } from '@angular/core';

import { ApiService } from './api.service';

@Component({

selector: 'app-component',

template: `

<form (submit)="sendEmail()">

<input type="email" name="to" [(ngModel)]="toEmail" required>

<input type="text" name="subject" [(ngModel)]="subject" required>

<textarea name="body" [(ngModel)]="body" required></textarea>

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

</form>

`,

})

export class AppComponent {

toEmail = '';

subject = '';

body = '';

constructor(private apiService: ApiService) {}

sendEmail() {

const formData = { to: this.toEmail, subject: this.subject, body: this.body };

this.apiService.sendEmail(formData).subscribe((response) => {

console.log('Email sent', response);

// Display a success message to the user

});

}

}

On the backend side (Spring Boot), create a REST endpoint to handle the email sending request. You can use the Spring MVC annotations to define the endpoint and the request type. Note that you will also need to inject the JavaMailSender bean to send the email.

@RestController

public class EmailController {

@Autowired

private JavaMailSender javaMailSender;

@PostMapping("/send-email")

public void sendEmail(@RequestBody EmailEntity emailEntity) throws MessagingException {

MimeMessage message = javaMailSender.createMimeMessage();

MimeMessageHelper helper = new MimeMessageHelper(message);

helper.setTo(emailEntity.getTo());

helper.setSubject(emailEntity.getSubject());

helper.setText(emailEntity.getBody(), true);

javaMailSender.send(message);

}

}

Define the EmailEntity class to represent the email data in the request body.

public class EmailEntity {

private String to;

private String subject;

private String body;

// getters and setters

}

Configure the email server properties in your Spring Boot application.properties file. You can use properties such as spring.mail.host , spring.mail.port , spring.mail.username , and spring.mail.password . These properties will be used by the JavaMailSender bean to connect to the email server.

spring.mail.host=smtp.gmail.com

spring.mail.port=587

spring.mail.username=<your-email>

spring.mail.password=<your-password>

spring.mail.properties.mail.smtp.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

That's it! With these steps, you have implemented email sending functionality using Spring Boot and Angular.

We apologize, we are experiencing a sharp increase in the number of requests at the moment.

We are working to increase our capacity to handle the load.

Sorry, you have reached your daily limit. Your limit will be updated in 23:45:32

If you dont want to wait, download our free app for Iphone or Android

Send