

**Module Introduction:**

Table of Contents

[1 Add file picker 1](#_Toc51585642)

[2 Convert template form to reactive form 2](#_Toc51585643)

[3 Add image Preview 5](#_Toc51585644)

[4 Front end validation – MIME type 8](#_Toc51585645)

[5 Server side logic to handle uploads 10](#_Toc51585646)

[5.1 Update Posts 10](#_Toc51585647)

[5.3 Fetch posts 15](#_Toc51585648)

[5.4 Fetching images on frontend: 16](#_Toc51585649)

# 1 Add file picker

Let’s go ahead and add an image picker to our component, Navigate to post-create.component.html and this code

<div>

<button mat-button>Pick Image</button>

</div>

Save, and try to validate, You can see if you click on it, all the validation rules are fired, because it tries to submit the form by default.

To change this set the type of button to Button, because the default if we don’t specify it will be submit.

<div>

<button mat-button type="button">Pick Image</button>

</div>

Changing to mat-stroked-button so that it will have outline, also let’s add some css to input type so that it is hidden when user clicks.

Input [type="file"] {

Visibility: hidden;

}

Let’s add a local reference #filePicker, so that whenever user selects the file the click event is fired.

<Div>

<button mat-stroked-button type="button" (click)="filePicker.click()">Pick Image</button>

<input type="file" #filePicker (change)="onImagePicked($event)">

</div>

After this, we need to register this input to our angular form, let’s go to our original form and try to convert the existing form template to reactive so that we can add more validations like, only upload image files, pdf etc.

# 2 Convert template form to reactive form

To use Reactive forms module, navigate to app.module.ts

Add

import { ReactiveFormsModule } from "@angular/forms";

under @Ngmodule imports,include

ReactiveFormsModule,

This will disable our ngmodel , navigate to the post-create.component.html and save , you can see errors returned.

We need to fix these, try removing all the ng references & arguments for controls and start removing the validations.

After this, we need to inject the form to the controller -postCreateComponent, to do this

Import FormGroup

import { NgForm, FormGroup } from '@angular/forms';

you can remove NgForm since it is obsolete now.

and under PostCreateComponent constructor, include

form: FormGroup;

export class PostCreateComponent implements OnInit {

enteredTitle = "";

enteredContent = "";

post: Post;

isLoading = false;

**form: FormGroup;**

imagePreview: string;

private mode = "create";

private postId: string;

FormGroup is the main form which has sub forms, form controls.

We need to initialize the form, this can be done in ngOnIt of PostCreateComponent

ngOnInit() {

this.form = new FormGroup({

title: new FormControl(null, {

validators: [Validators.required, Validators.minLength(3)]

}),

content: new FormControl(null, { validators: [Validators.required] }),

});

After this , we need to set Initial values to our form we can do it under ngOnInit,

this.route.paramMap.subscribe((paramMap: ParamMap) => {

if (paramMap.has("postId")) {

this.mode = "edit";

this.postId = paramMap.get("postId");

this.isLoading = true;

this.postsService.getPost(this.postId).subscribe(postData => {` `

this.isLoading = false;

this.post = {

id: postData.\_id,

title: postData.title,

content: postData.content

};

this.form.setValue({

title: this.post.title,

content: this.post.content

});

Now when we save our post , we will not have our form as argument , instead we can access the reference. Navigate to OnSavePost

onSavePost() {

if (this.form.invalid) {

return;

}

onSavePost() {

if (this.form.invalid) {

return;

}

this.isLoading = true;

if (this.mode === "create") {

this.postsService.addPost(

this.form.value.title,

this.form.value.content

);

} else {

this.postsService.updatePost(

this.postId,

this.form.value.title,

this.form.value.content

);

}

this.form.reset();

}

}

Now we need to sync our html with our typescript code,

To do this , navigate to post-create Component.html and change the directives under forms as below

<form [formGroup]="form" (submit)="onSavePost()" \*ngIf="!isLoading">

<mat-form-field>

<input matInput type="text" formControlName="title" placeholder="Post Title">

<mat-error \*ngIf="form.get('title').invalid">Please enter a post title.</mat-error>

</mat-form-field>

<div>

<button mat-stroked-button type="button" (click)="filePicker.click()">Pick Image</button>

<input type="file" #filePicker >

</div>

<mat-form-field>

<textarea matInput rows="4" formControlName="content" placeholder="Post Content"></textarea>

<mat-error \*ngIf="form.get('content').invalid">Please enter a post title.</mat-error>

</mat-form-field>

<button mat-raised-button color="accent" type="submit">Save Post</button>

</form>

# Add image Preview

**Adding Image control to store the image:**

To do this, let’s add change event on file picker, as soon as you select the file the event will pick the file

<input type="file" #filePicker (change)="onImagePicked($event)">

Let’s go to our typescript code and add onImagePicked method as below ,

Typescript code cannot access the files object since it doesn’t understand the component ,

const file = (event.target.files)

}

Change it to as below and also we need to add this file to a form control, navigate to our ngOnInit

ngOnInit() {

this.form = new FormGroup({

title: new FormControl(null, {

validators: [Validators.required, Validators.minLength(3)]

}),

content: new FormControl(null, { validators: [Validators.required] }),

image: new FormControl(null, {

validators: [Validators.required],

asyncValidators: [mimeType]

})

});

Also , we don’t want to bind this image on our html template, reason being we don’t want to synchronize the field to html, this is feature of using reactive forms ,you can control the field in your typescript ,take this value and process it accordingly.

Navigate to your onImagePicked event and try to access the form

this.form.patchValue() targets only updating one field not all fields.

this.form.get("image").updateValueAndValidity();

this is how we pick the image from the form and Inform angular I changed value and hence revalidate &store the value and ensure the value passed is valid.

Add some console logs to validate if the form & file are present. Eventually, the function will be as below.

onImagePicked(event: Event) {

const file = (event.target as HTMLInputElement).files[0];

this.form.patchValue({ image: file });

this.form.get("image").updateValueAndValidity();

}

Next let’s work on previewing the Image for the User.

Navigate to post-create.component.html and add a new control after filepicker,

Also we want to ensure the imagePreview is stored as a url and is binded to our src property.

To do this , create a imagePreview property in PostCreateComponent

export class PostCreateComponent implements OnInit {

enteredTitle = "";

enteredContent = "";

post: Post;

isLoading = false;

form: FormGroup;

imagePreview: string;

and to access the file as url , we need to store the file result as string into imagePreview

onImagePicked(event: Event) {

const file = (event.target as HTMLInputElement).files[0];

this.form.patchValue({ image: file });

this.form.get("image").updateValueAndValidity();

const reader = new FileReader();

reader.onload = () => {

this.imagePreview = reader.result as string;

};

reader.readAsDataURL(file);

}

Let’s bind this to UI and we have ngif because we don’t want to show preview when it is null or empty.

Let’s also add some css to our post-create component,

.image-preview {

height: 5rem;

margin: 1rem 0;

}

.image-preview img {

height: 100%;

}

And eventually apply it

<div class="image-preview" \*ngIf="imagePreview !== '' && imagePreview

">

<img [src]="imagePreview" [alt]="form.value.title">

</div>

# Front end validation – MIME type

**MIME Type validator:**

Let’s now add feature to validate our upload, like which all MIME types are allowed.

Under Post-create folder, create a file mime-type.validator.ts

import { AbstractControl } from "@angular/forms";

import { Observable, Observer} from "rxjs";

export const mimeType = (

control: AbstractControl

): Promise<{ [key: string]: any }> | Observable<{ [key: string]: any }> => {

const file = control.value as File;

const fileReader = new FileReader();

const frObs = Observable.create(

(observer: Observer<{ [key: string]: any }>) => {

fileReader.addEventListener("loadend", () => {

const arr = new Uint8Array(fileReader.result as ArrayBuffer).subarray(0, 4);

let header = "";

let isValid = false;

for (let i = 0; i < arr.length; i++) {

header += arr[i].toString(16);

}

switch (header) {

case "89504e47":

isValid = true;

break;

case "ffd8ffe0":

case "ffd8ffe1":

case "ffd8ffe2":

case "ffd8ffe3":

case "ffd8ffe8":

isValid = true;

break;

default:

isValid = false; // Or you can use the blob.type as fallback

break;

}

if (isValid) {

observer.next(null);

} else {

observer.next({ invalidMimeType: true });

}

observer.complete();

});

fileReader.readAsArrayBuffer(file);

}

);

return frObs;

};

Once created , we need to attach it to the typescript code.post-create.component.ts

import { mimeType } from "./mime-type.validator";

ngOnInit() {

this.form = new FormGroup({

title: new FormControl(null, {

validators: [Validators.required, Validators.minLength(3)]

}),

content: new FormControl(null, { validators: [Validators.required] }),

image: new FormControl(null, {

validators: [Validators.required],

asyncValidators: [mimeType]

})

});

Also, let’s fix the validator , it shouldn’t show preview if the mime type is not valid .Navigate to post-create.component.ts

<div class="image-preview" \*ngIf="imagePreview !== '' && imagePreview && form.get('image').valid">

<img [src]="imagePreview" [alt]="form.value.title">

</div>

# Server side logic to handle uploads

## Update Posts

The current body parser can only be used for parsing json /url inputs. To transform file input, we need another npm package ‘multer’ to extract incoming files.

npm install –save multer

navigate to backend code, routes/posts.js

const express = require("express");

const multer = require("multer");

const Post = require("../models/post");

const router = express.Router();

const MIME\_TYPE\_MAP = {

"image/png": "png",

"image/jpeg": "jpg",

"image/jpg": "jpg"

};

const storage = multer.diskStorage({

destination: (req, file, cb) => {

const isValid = MIME\_TYPE\_MAP[file.mimetype];

let error = new Error("Invalid mime type");

if (isValid) {

error = null;

}

cb(error, "backend/images");

},

filename: (req, file, cb) => {

const name = file.originalname

.toLowerCase()

.split(" ")

.join("-");

const ext = MIME\_TYPE\_MAP[file.mimetype];

cb(null, name + "-" + Date.now() + "." + ext);

}

});

router.post('', multer({storage: storage}).single("image"), (req, res, next) => {

const post = new Post({

title: req.body.title,

content: req.body.content

});

post.save()

.then(createdPost => {

res.status(201).json({

message: 'Post Added !!',

postId: createdPost.\_id

});

});

});

module.exports = router;

Let’s go back to our front end angular code – services

We need to ensure when we add the file we send the right content, form data in this case because with json we cannot send the blob. But with form data, we can send both tex values as well as blob.

Navigate to posts-service.ts and update your add post function as below,

addPost(title: string, content: string, image: File) {

const postData = new FormData();

postData.append("title", title);

postData.append("content", content);

postData.append("image", image, title);

this.http

.post<{ message: string; postId: string }>(

"http://localhost:3000/api/posts",

postData

)

.subscribe(responseData => {

const post: Post = {

id: responseData.postId,

title: title,

content: content,

};

this.posts.push(post);

this.postsUpdated.next([...this.posts]);

this.router.navigate(["/"]);

});

}

And in post-create component, onSavePost we need to send the image as well along with request.

onSavePost() {

if (this.form.invalid) {

return;

}

this.isLoading = true;

if (this.mode === "create") {

this.postsService.addPost(

this.form.value.title,

this.form.value.content,

this.form.value.image

);

}

Validate the MIME Type on server by disabling the angular validation.

Next topic is to pass back file info to client that we stored in database and later can update files.

We need to ensure there is an image path in our database on what is the file about and send it back to user in case he wants to update them.

Navigate to our backend code and our posts model and add new property as below.

const mongoose = require("mongoose");

const postSchema = mongoose.Schema({

title: { type: String, required: true },

content: { type: String, required: true },

imagePath: { type: String, required: true }

});

module.exports = mongoose.model("Post", postSchema);

Now navigate to routes/posts.js , we need to construct the imagePath via url.

router.post(

"",

multer({ storage: storage }).single("image"),

(req, res, next) => {

const url = req.protocol + "://" + req.get("host");

const post = new Post({

title: req.body.title,

content: req.body.content,

imagePath: url + "/images/" + req.file.filename

});

post.save().then(createdPost => {

res.status(201).json({

message: "Post added successfully",

post: {

...createdPost,

id: createdPost.\_id

}

});

});

}

);

Navigate to the front end code, post.model.ts and add the new property imagePath

export interface Post {

id: string;

title: string;

content: string;

imagePath: string;

}

And in posts.service.ts,

In the below Add Post function ,Change the return type of http post call from postId to post object since we are returning more fields now.

addPost(title: string, content: string, image: File) {

const postData = new FormData();

postData.append("title", title);

postData.append("content", content);

postData.append("image", image, title);

this.http

.post<{ message: string; post: Post }>(

"http://localhost:3000/api/posts",

postData

)

.subscribe(responseData => {

const post: Post = {

id: responseData.post.id,

title: title,

content: content,

imagePath: responseData.post.imagePath

};

this.posts.push(post);

this.postsUpdated.next([...this.posts]);

this.router.navigate(["/"]);

});

}

update the UpdatePost in post.service.ts with the imagePath , for now set it to null.

Also update the ngonit in post.create.component.ts

ngOnInit() {

this.form = new FormGroup({

title: new FormControl(null, {

validators: [Validators.required, Validators.minLength(3)]

}),

content: new FormControl(null, { validators: [Validators.required] }),

image: new FormControl(null, {

validators: [Validators.required],

asyncValidators: [mimeType]

})

});

this.route.paramMap.subscribe((paramMap: ParamMap) => {

if (paramMap.has("postId")) {

this.mode = "edit";

this.postId = paramMap.get("postId");

this.isLoading = true;

this.postsService.getPost(this.postId).subscribe(postData => {

this.isLoading = false;

this.post = {

id: postData.\_id,

title: postData.title,

content: postData.content,

imagePath: null

};

this.form.setValue({

title: this.post.title,

content: this.post.content,

});

## **Fetch posts**

Navigate to getPosts function of posts.service.ts and include imagePath in the return

getPosts() {

this.http

.get<{ message: string; posts: any }>("http://localhost:3000/api/posts")

.pipe(

map(postData => {

return postData.posts.map(post => {

return {

title: post.title,

content: post.content,

id: post.\_id,

imagePath: post.imagePath

};

});

})

)

.subscribe(transformedPosts => {

this.posts = transformedPosts;

this.postsUpdated.next([...this.posts]);

});

}

## **Fetching images on frontend**:

Let’s work now to render the images while we fetch the list of posts,

Navigate to post.list.ts component

And create a new div with the new image field.

<div class="post-image">

<img [src]="post.imagePath" [alt]="post.title">

</div>

Apply css for post-image

.post-image {

width: 100%;

}

.post-image img {

width: 100%;

box-shadow: 1px 1px 5px rgba(0, 0, 0, 0.5);

}

If you notice, all our earlier posts , didn’t had the imagePath that’s the reason why they are not visible , delete all the posts and try new posts. You will notice still we get a 404 in the network tab in developer tools.

This is because the express server doesn’t allow to access files under /images by default.

To access them, navigate to your app.js file of backend code and add after the bodyparser. This will give access to your static files. Any request targeted to access these /images folder will be given access.

app.use("/images", express.static(path.join("backend/images")));

also require this path module in the top.

const path = require("path");

Now, try to run the app and add post and list them. You should see all the posts with images attached properly.

**Update Posts:**

We wanted to edit images for viewing more than once. We need to work on post-create-component.ts

Since earlier when we landed to the post create component, we had null in the image path .Now since we will be having posts /image path already . Lets update ngonIt as highlighted below to include imagePath

ngOnInit() {

this.form = new FormGroup({

title: new FormControl(null, {

validators: [Validators.required, Validators.minLength(3)]

}),

content: new FormControl(null, { validators: [Validators.required] }),

image: new FormControl(null, {

validators: [Validators.required],

asyncValidators: [mimeType]

})

});

this.route.paramMap.subscribe((paramMap: ParamMap) => {

if (paramMap.has("postId")) {

this.mode = "edit";

this.postId = paramMap.get("postId");

this.isLoading = true;

this.postsService.getPost(this.postId).subscribe(postData => {

this.isLoading = false;

this.post = {

id: postData.\_id,

title: postData.title,

content: postData.content,

imagePath: null

};

this.form.setValue({

title: this.post.title,

content: this.post.content,

**image: this.post.imagePath**

});

});

} else {

this.mode = "create";

this.postId = null;

}

});

}

Also ensure onSavePost, we are sending the image as below. In the updatePost function call as below

onSavePost() {

if (this.form.invalid) {

return;

}

this.isLoading = true;

if (this.mode === "create") {

this.postsService.addPost(

this.form.value.title,

this.form.value.content,

this.form.value.image

);

} else {

this.postsService.updatePost(

this.postId,

this.form.value.title,

this.form.value.content,

this.form.value.image

);

}

this.form.reset();

}

}

Now Navigate to post-service.ts to your updatePost function , to include the image and check if it is object or json

updatePost(id: string, title: string, content: string, image: File | string) {

let postData: Post | FormData;

if (typeof image === "object") {

postData = new FormData();

postData.append("title", title);

postData.append("content", content);

postData.append("image", image, title);

} else {

postData = {

id: id,

title: title,

content: content,

imagePath: image

};

}

this.http

.put("http://localhost:3000/api/posts/" + id, postData)

.subscribe(response => {

const updatedPosts = [...this.posts];

const oldPostIndex = updatedPosts.findIndex(p => p.id === id);

const post: Post = {

id: id,

title: title,

content: content,

imagePath: response.imagePath

};

updatedPosts[oldPostIndex] = post;

this.posts = updatedPosts;

this.postsUpdated.next([...this.posts]);

this.router.navigate(["/"]);

});

}

We need to fix the response.imagePath because the current update /put operation on server /backend doesn’t send this . For temporary, set the imagePath as “” till we fix.

imagePath: “”

Navigate to backend routes/posts.js

Go to put function and replace the existing with below

router.put(

"/:id",

multer({ storage: storage }).single("image"),

(req, res, next) => {

const post = new Post({

\_id: req.body.id,

title: req.body.title,

content: req.body.content,

});

console.log(post);

Post.updateOne({ \_id: req.params.id }, post).then(result => {

res.status(200).json({ message: "Update successful!" });

});

}

);

Now when we try to save the post, it will fail because there is a validation onSavePost() in post.create.componet.ts , which checks if the image MIMETYpe is valid but in here we are sending a string /url of the file. Let’s fix it.

Navigate to our MIME Type validator

And add this logic to the MIME Type validator to check if its string return an observable null

if (typeof(control.value) === 'string') {

return of(null);

}

import { Observable, Observer, of } from "rxjs";

import { AbstractControl } from "@angular/forms";

import { Observable, Observer, of } from "rxjs";

export const mimeType = (

control: AbstractControl

): Promise<{ [key: string]: any }> | Observable<{ [key: string]: any }> => {

if (typeof(control.value) === 'string') {

return of(null);

}

const file = control.value as File;

const fileReader = new FileReader();

const frObs = Observable.create(

(observer: Observer<{ [key: string]: any }>) => {

fileReader.addEventListener("loadend", () => {

const arr = new Uint8Array(fileReader.result as ArrayBuffer).subarray(0, 4);

let header = "";

let isValid = false;

for (let i = 0; i < arr.length; i++) {

header += arr[i].toString(16);

}

switch (header) {

case "89504e47":

isValid = true;

break;

case "ffd8ffe0":

case "ffd8ffe1":

case "ffd8ffe2":

case "ffd8ffe3":

case "ffd8ffe8":

isValid = true;

break;

default:

isValid = false; // Or you can use the blob.type as fallback

break;

}

if (isValid) {

observer.next(null);

} else {

observer.next({ invalidMimeType: true });

}

observer.complete();

});

fileReader.readAsArrayBuffer(file);

}

);

return frObs;

};

Even now ,when we try to save it fails because , when we initialize the first time on ngInit of postCreateComponent, we do a getPost and it doesn’t have imagePath as another parameter in response , add it as below. Navigate to Posts.Service.ts , getPost function and ass as below

getPost(id: string) {

return this.http.get<{ \_id: string, title: string, content: string, imagePath: string }>(

"http://localhost:3000/api/posts/" + id

);

}

Now ,navigate to your ngOnit of post-create component and include the response of imagePath from getPost.

imagePath: postData.imagePath

ngOnInit() {

this.form = new FormGroup({

title: new FormControl(null, {

validators: [Validators.required, Validators.minLength(3)]

}),

content: new FormControl(null, { validators: [Validators.required] }),

image: new FormControl(null, {

validators: [Validators.required],

asyncValidators: [mimeType]

})

});

this.route.paramMap.subscribe((paramMap: ParamMap) => {

if (paramMap.has("postId")) {

this.mode = "edit";

this.postId = paramMap.get("postId");

this.isLoading = true;

this.postsService.getPost(this.postId).subscribe(postData => {

this.isLoading = false;

this.post = {

id: postData.\_id,

title: postData.title,

content: postData.content,

imagePath: postData.imagePath

};

this.form.setValue({

title: this.post.title,

content: this.post.content,

image: this.post.imagePath

});

});

} else {

this.mode = "create";

this.postId = null;

}

});

}

Now you can save a post , but let’s go back to our backend code, routes/posts.js

PUT function to add a condition to check if the request.file is present then pick from the Image path where file already exists else create new post , update as below

router.put(

"/:id",

multer({ storage: storage }).single("image"),

(req, res, next) => {

let imagePath = req.body.imagePath;

if (req.file) {

const url = req.protocol + "://" + req.get("host");

imagePath = url + "/images/" + req.file.filename

}

const post = new Post({

\_id: req.body.id,

title: req.body.title,

content: req.body.content,

imagePath: imagePath

});

console.log(post);

Post.updateOne({ \_id: req.params.id }, post).then(result => {

res.status(200).json({ message: "Update successful!" });

});

}

);

Now try to edit the post and update the image use the same image , the upload fails because we didn’t send the id filed in our posts.service, update posts function. Append the missing id into postData as below.

Also ensure, you update instead of post.id , use id.

**const oldPostIndex = updatedPosts.findIndex(p => p.id === id);**

postData.append("id", id);

updatePost(id: string, title: string, content: string, image: File | string) {

let postData: Post | FormData;

if (typeof image === "object") {

postData = new FormData();

postData.append("id", id);

postData.append("title", title);

postData.append("content", content);

postData.append("image", image, title);

} else {

postData = {

id: id,

title: title,

content: content,

imagePath: image

};

}

this.http

.put("http://localhost:3000/api/posts/" + id, postData)

.subscribe(response => {

const updatedPosts = [...this.posts];

**const oldPostIndex = updatedPosts.findIndex(p => p.id === id);**

const post: Post = {

id: id,

title: title,

content: content,

imagePath: ""

};

updatedPosts[oldPostIndex] = post;

this.posts = updatedPosts;

this.postsUpdated.next([...this.posts]);

this.router.navigate(["/"]);

});

}

Now try to edit the image and save. It should save it and show us the list of posts.

We achieved a lot, added upload, file picker, added MIME type validation both on UI & backend and include the update/edit functionality for the image uploads as well.