Single Image Upload

Angular

**document-upload.component.html**

|  |
| --- |
| <div class="container"> |
|  | <div class="row"> |
|  | <div class="col s8 offset-2"> |
|  | <div class="card-panel teal lighten-2"> |
|  | <label style="color:white">Upload your ID proof:</label> |
|  | </div> |
|  | <form #imageForm=ngForm (ngSubmit)="uploadFileToActivity(imageForm)"> |
|  | <div class="row"> |
|  | <div class="input-field col s12"> |
|  | <!-- <input type="text" required #username ngModel id="username" name = "username" class = "form-control"> --> |
|  | <!-- <input type="text" #username ngModel name="username" id="username" required> --> |
|  | <!-- <label for="username">User Name</label> --> |
|  | </div> |
|  | </div> |
|  | <input type="file" #Image id="file" accept="image/\*" style="color: white;" (change)="handleFileInput($event)"> |
|  | <button type="submit" class="btn btn-primary" [disabled]="!imageForm.valid">Upload</button> |
|  | </form> |
|  | </div> |
|  | </div> |
|  | </div> |

## ****document-upload.component.ts****

|  |
| --- |
| import { Component, OnInit } from '@angular/core'; |
|  | import { NgForm } from '@angular/forms'; |
|  | import { Dbimg } from 'src/app/models/Dbimg'; |
|  | import { UploadImageService } from 'src/app/services/upload-image.service'; |
|  | //import \* as FileSaver from 'file-saver'; |
|  |  |
|  | @Component({ |
|  | selector: 'app-document-upload', |
|  | templateUrl: './document-upload.component.html', |
|  | styleUrls: ['./document-upload.component.css'] |
|  | }) |
|  | export class DocumentUploadComponent implements OnInit { |
|  |  |
|  | db:Dbimg= new Dbimg(0,"",""); |
|  | constructor(private imageService : UploadImageService) { } |
|  | fileToUpload: any; |
|  | document:string=""; |
|  | blob:any; |
|  | reader:any; |
|  | url:any; |
|  | ngOnInit(): void { |
|  |  |
|  | } |
|  | handleFileInput(event:any) { |
|  | this.fileToUpload = event.target.files.item(0); |
|  | this.reader = new FileReader(); |
|  | this.reader.readAsDataURL(this.fileToUpload); |
|  | this.reader.onload = () => { |
|  | this.blob = new Blob(event.target.files, { type: event.target.files[0].type }); |
|  | this.url = window.URL.createObjectURL(this.blob); |
|  | open(this.url) |
|  | console.log(typeof(this.blob)); |
|  | console.log(this.url); |
|  | }; |
|  | } |
|  |  |
|  |  |
|  |  |
|  | uploadFileToActivity(imgForm:NgForm) { |
|  | if(this.fileToUpload!=null ){ |
|  | this.db.ImageFile=this.url; |
|  | //FileSaver.saveAs(this.blob,this.db.ImageFile); |
|  | this.db.ImageName=imgForm.value.username; |
|  | console.log(this.db); |
|  | this.imageService.postFile(this.db).subscribe(data => { |
|  | // do something, if upload success |
|  | }, error => { |
|  | console.log(error); |
|  | }); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | } |

API

**Model**

**DocumentsController**

|  |
| --- |
| [Route("api/[controller]")] |
|  | [ApiController] |
|  |  |
|  | public class DocumentsController : ControllerBase |
|  | { |
|  |  |
|  |  |
|  | private readonly ProjectGladiatorContext \_context; |
|  |  |
|  | public DocumentsController(ProjectGladiatorContext context) |
|  | { |
|  | \_context = context; |
|  | } |
|  |  |
|  | // GET: api/Documents |
|  | [HttpGet] |
|  | public async Task<ActionResult<IEnumerable<Document>>> GetDocuments() |
|  | { |
|  | return await \_context.Documents.ToListAsync(); |
|  | } |
|  |  |
|  | // GET: api/Documents/5 |
|  | [HttpGet("{id}")] |
|  | public async Task<ActionResult<Document>> GetDocument(int id) |
|  | { |
|  | var document = await \_context.Documents.FindAsync(id); |
|  |  |
|  | if (document == null) |
|  | { |
|  | return NotFound(); |
|  | } |
|  |  |
|  | return document; |
|  | } |
|  |  |
|  | // PUT: api/Documents/5 |
|  | // To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754 |
|  | [HttpPut("{id}")] |
|  | public async Task<IActionResult> PutDocument(int id, Document document) |
|  | { |
|  | if (id != document.DocumentId) |
|  | { |
|  | return BadRequest(); |
|  | } |
|  |  |
|  | \_context.Entry(document).State = EntityState.Modified; |
|  |  |
|  | try |
|  | { |
|  | await \_context.SaveChangesAsync(); |
|  | } |
|  | catch (DbUpdateConcurrencyException) |
|  | { |
|  | if (!DocumentExists(id)) |
|  | { |
|  | return NotFound(); |
|  | } |
|  | else |
|  | { |
|  | throw; |
|  | } |
|  | } |
|  |  |
|  | return NoContent(); |
|  | } |
|  |  |
|  | // POST: api/Documents |
|  | // To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754 |
|  | /\* [HttpPost] |
|  | public async Task<ActionResult<Document>> PostDocument(Document document) |
|  | { |
|  |  |
|  |  |
|  | \_context.Documents.Add(document); |
|  | await \_context.SaveChangesAsync(); |
|  |  |
|  | return CreatedAtAction("GetDocument", new { id = document.DocumentId }, document); |
|  | }\*/ |
|  |  |
|  | // DELETE: api/Documents/5 |
|  | [HttpDelete("{id}")] |
|  | public async Task<IActionResult> DeleteDocument(int id) |
|  | { |
|  | var document = await \_context.Documents.FindAsync(id); |
|  | if (document == null) |
|  | { |
|  | return NotFound(); |
|  | } |
|  |  |
|  | \_context.Documents.Remove(document); |
|  | await \_context.SaveChangesAsync(); |
|  |  |
|  | return NoContent(); |
|  | } |
|  |  |
|  | private bool DocumentExists(int id) |
|  | { |
|  | return \_context.Documents.Any(e => e.DocumentId == id); |
|  | } |
|  | [HttpPost] |
|  | public IActionResult UploadImage(IFormFile file) |
|  | { |
|  |  |
|  | if (file.Length > 0) |
|  | { |
|  | var fileName = Path.GetFileName(file.FileName); |
|  | using (FileStream stream = new FileStream(Path.Combine("Image", fileName), FileMode.Create)) |
|  |  |
|  | file.CopyTo(stream); |
|  | Document doc = new Document(); |
|  | doc.Username = fileName; |
|  |  |
|  | \_context.Documents.Add(doc); |
|  | \_context.SaveChanges(); |
|  |  |
|  | } |
|  | return Ok(); |
|  | } |
|  | } |
|  | } |