DS3103 Webutvikling

Rolando Gonzalez
2022

Bildeopplast Web Api

Innhold

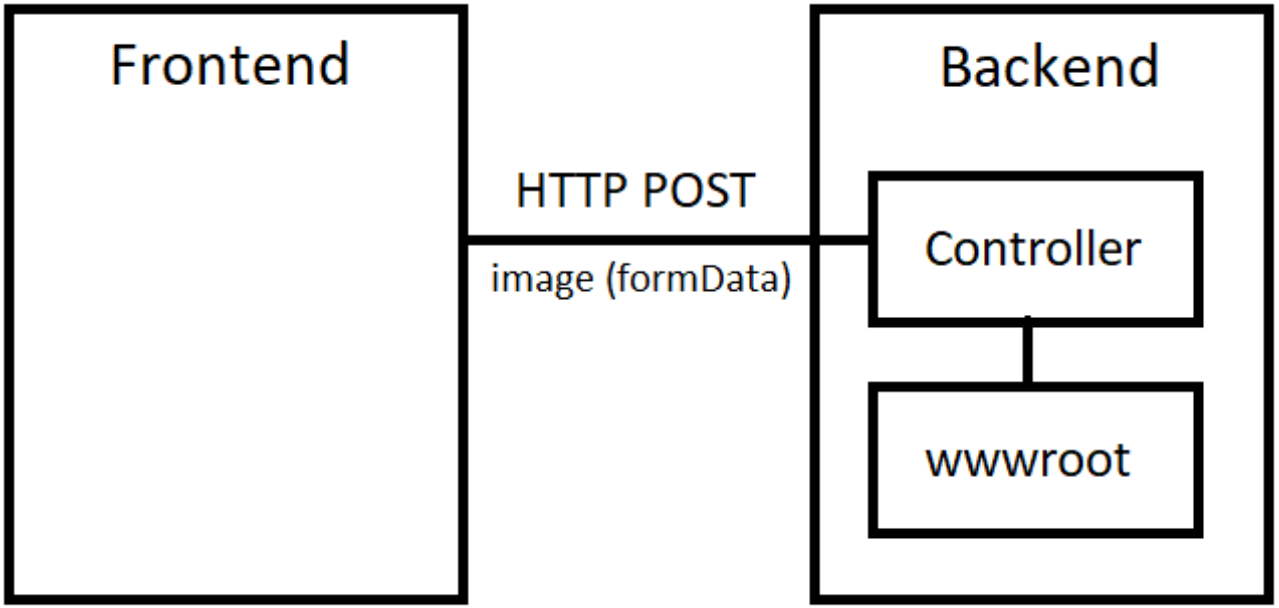
- Hva er bildeopplast?
- Hvordan skjer bildeopplast?
- Teknisk forklaring av bildeopplast
- Koden for bildeopplast i Controller (backend)
- Koden for bildeopplast i React component
- Koden for bildeopplast i en Service (frontend)

Hva er bildeopplast?

- Bildeopplast dreier seg om å la bruker kunne laste opp et bilde fra sin maskin opp til en backend-løsning.
- I et Web Api kan man benytte wwwroot-mappen som et lagringssted for bilder som brukere laster opp.

Hvordan skjer bildeopplast?

- Figuren viser at det fra frontend gjøres en HTTP Post request som inneholder bildet i form av et formData-objekt. Controller tar imot requesten og lagrer til wwwroot.



Teknisk forklaring av bildeopplast

- **Frontend:**
  - Man bruker <input type="file"/> for å la bruker velge bilde
  - Input-elementet har et attributt som heter «files» hvor det valgte bildeobjektet ligger
  - Før sending (http request Post) over til Controller i backend pakker man inn bildet i et formData-objekt

Teknisk forklaring av bildeopplast

- **Backend:**
  - Controller tar imot http request post og konverterer formData-objektet til et IFile-objekt
  - Bildet lagres i wwwroot-mappen ved hjelp av filstrøm (FileStream)

  - Man har gjerne 1 dedikert Controller for filopplasting i Web Apiet – det er ikke behov for å ha den samme koden i alle Controllere.

Koden for bildeopplast i Controller

- Man overlater ansvaret for å generere filsti til wwwroot til Web Apiet; filstien er egentlig en lengre filsti som inkluderer alt fra serverens rot inn i webapiet og til slutt inn i wwwroot-mappen.
- Ved hjelp av et FileStream-objekt lagres bildet.

C# FileUploadController.cs

ProductApi > Controllers > C# FileUploadController.cs > ...

```
1   using Microsoft.AspNetCore.Mvc;
2
3   namespace ProductApi.Controllers;
4
5   [ApiController]
6   [Route("[controller]")]
    0 references
7   public class FileUploadController : ControllerBase
8   {
        2 references
9       private readonly IWebHostEnvironment hosting;
10
        0 references
11      public FileUploadController(IWebHostEnvironment _hosting)
12      {
13          hosting = _hosting;
14      }
15
16      [HttpPost]
        0 references
17      public IActionResult SaveImage(IFormFile file)
18      {
19          string webRootPath = hosting.WebRootPath;
20          string absolutePath = Path.Combine($"{webRootPath}/images/{file.FileName}");
21
22          using(var fileStream = new FileStream(absolutePath, FileMode.Create))
23          {
24              file.CopyTo(fileStream);
25          }
26
27          return Ok();
28      }
29  }
```

Koden for bildeopplast i React component

- Får tak i files-attributtet fra input type file.
- files er et array
- files[0] er bildet som bruker velger

UploadProductImage.tsx

src > components > UploadProductImage.tsx > ...

```tsx
import {useState, ChangeEvent} from 'react';
import ImageUploadService from '../services/ImageUploadService';

const UploadProductImage = () => {

    const [image, setImage] = useState<File | null>(null);

    const setImageHandler = (event: ChangeEvent<HTMLInputElement>) => {
        const {files} = event.target;
        if(files != null){
            const file = files[0];
            setImage(file);
        }
    }

    const uploadImage = () => {
        if(image!= null){
            ImageUploadService.uploadImage(image);
        }
    }

    return (
        <section>
            <h3>Last opp bilde</h3>
            <label>Bilde</label>
            <input onChange={setImageHandler} type="file"/>
            <input onClick={uploadImage} type="button" value="Last opp bilde"/>
        </section>
    )
}
```

Koden for bildeopplast i Service

- Funksjonen tar imot en File fra en komponent som bruker servicen.
- Funksjonen pakker inn bildet i et FormData-objekt.
- POST har med konfigurasjon om at det er en multipart/form-data (noe et bilde er)

ImageUploadService.ts

```
1   import axios from "axios";
2
3   const ImageUploadService = (
4       ()=>{
5
6           const imageUploadEndpoint = "https://localhost:7125/uploadimage";
7
8           const uploadImage = async (image: File) => {
9               const formData = new FormData();
10              formData.append("file", image);
11
12              const result = await axios({
13                  url: imageUploadEndpoint,
14                  method: "POST",
15                  data: formData,
16                  headers: { "Content-Type": "multipart/form-data" }
17              });
18
19              formData.delete("file");
20          }
21
22          return {
23              uploadImage
24          }
25      }
26  )();
27
28  export default ImageUploadService;
```

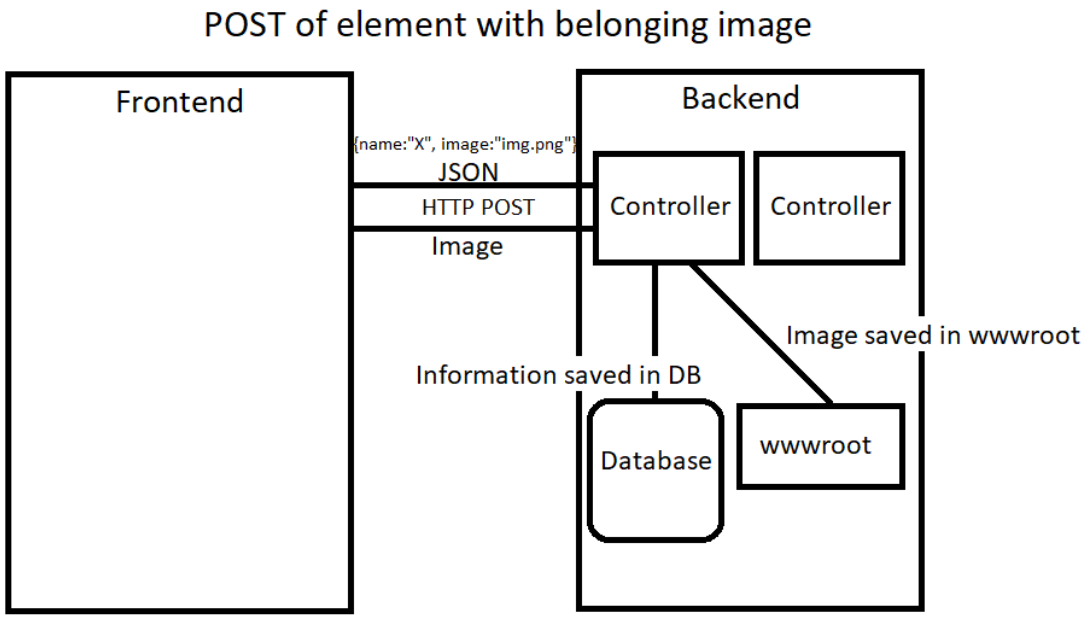
Angående result fra axios POST

- Man bør tenke over hva slags informasjon man skal gi til brukeren etter at bildeopplasten har skjedd.
- For eksempel bør man kanskje skrive ut til brukeren at det gikk bra å laste opp bildet, eller eventuelt at noe gikk galt ved bildeopplast.
- result-objektet fra forrige slide bør sjekkes for å styre hva som skal skje etter bildeopplast.

Om å laste opp objekt med tilhørende bilde

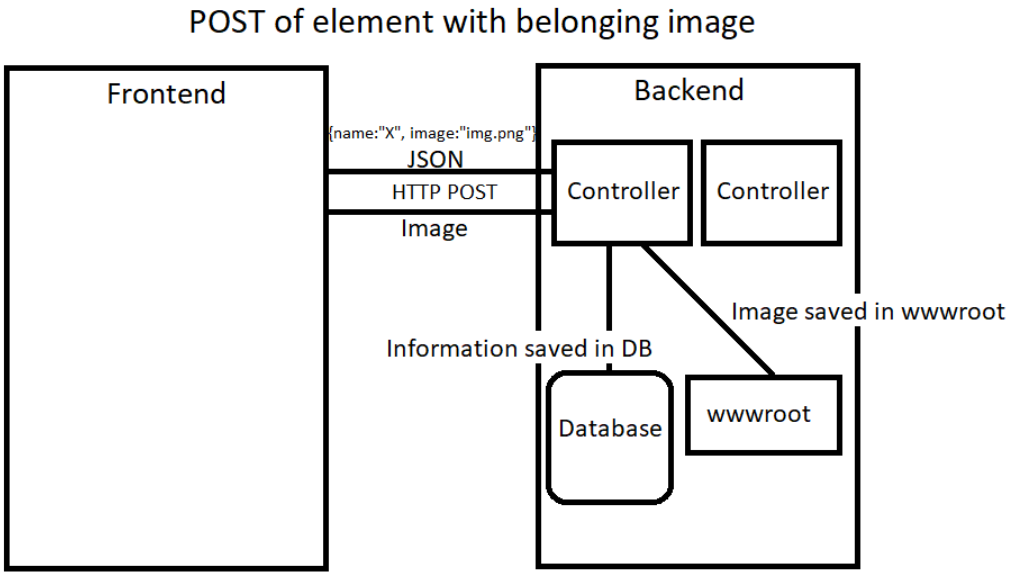
Om bildeopplast samtidig med objekt

- Man ønsker ofte å laste opp både objekt av noe og tilhørende bilde. Eksempel kan være at man registrerer en produkt med navn, pris osv. og samtidig laster opp et tilhørende produktbilde.



Om bildeopplast samtidig med objekt

- Man vil da gjøre 2 http request etter hverandre: 1 til controller for å lagre objektet og 1 til controller for å lagre bildet i wwwroot.
- Objektet i frontend, som ender opp med å lagres i databasen, vil inneholde navnet på bildet:




```
{
  name: "Samsung TV 72",
  image: "samsung-tv-72.png"
}
```

Om bildeopplast samtidig med objekt

- Når man henter et produkt fra Web Api fra frontend vil man få objektet som inneholder navn og bildenavn og man vil kunne vise det ved å referere til Web Apiets url til wwwroot -> images-mappe

****