Code Academy

Lecturer

Rokas Slaboševičius

# Working with photos and validating requests

Data

# Today you will learn

01 How do I upload a photo?

02 How do I download a photo from db?

03 How do I adjust the properties of an uploaded photo?

# We create an object that we will use to accept the photo upload

First we need to make sure that the uploaded object is of the image file type and that it matches our desired size

```
public class ImageUploadRequest
{
    public IFormFile Image { get; set; }
}
```

# Request validation

Two attributes can be created to validate the file size and type.

The first attribute will be responsible for the size and the second for the type

# MaxFileSizeAttribute

```
public class MaxFileSizeAttribute : ValidationAttribute
{
    private readonly int _maxFileSize;
    public MaxFileSizeAttribute(int maxFileSize)
    {
        _maxFileSize = maxFileSize;
    }

    protected override ValidationResult IsValid(
    object value, ValidationContext validationContext)
    {
        if (value is IFormFile file)
        {
            if (file.Length > _maxFileSize)
            {
                return new ValidationResult(GetErrorMessage());
            }
        }

        return ValidationResult.Success;
    }

    public string GetErrorMessage()
    {
        return $"Maximum allowed file size is { _maxFileSize} bytes.";
    }

}
```

# AllowedExtensionsAttribute

```
public class AllowedExtensionsAttribute : ValidationAttribute
{
    private readonly string[] _extensions;
    public AllowedExtensionsAttribute(string[] extensions)
    {
        _extensions = extensions;
    }

    protected override ValidationResult IsValid(
    object value, ValidationContext validationContext)
    {
        if (value is IFormFile file)
        {
            var extension = Path.GetExtension(file.FileName);
            if (!_extensions.Contains(extension.ToLower()))
            {
                return new ValidationResult(GetErrorMessage());
            }
        }

        return ValidationResult.Success;
    }

    public string GetErrorMessage()
    {
        return $"This photo extension is not allowed!";
    }
}
```

# After applying the attributes, the class now looks like this

The file size is adjustable by yourself, and there are also more extension types to choose from.

```
public class ImageUploadRequest
{
    [MaxFileSize(5 * 1024 * 1024)]
    [AllowedExtensions(new string[] {".png",".jpg"})]
    public IFormFile Image { get; set; }
}
```

# Upload endpoint

Endpoint would look like this, it would be worth considering moving the conversion to bits to a service that would call the Save repository method after the conversion

```
[ApiController]
[Route("[controller]")]
public class ImageController : ControllerBase
{
    [HttpPost("Upload")]
    public ActionResult UploadImage([FromForm] ImageUploadRequest imageRequest)
    {
        using var memoryStream = new MemoryStream();
        imageRequest.Image.CopyTo(memoryStream);
        // you can save this to database now
        var imageBytes = memoryStream.ToArray();
        return Ok();
    }
}
```

# Download endpoint

The download endpoint should pick up the bits from the database and return them as File

We start by creating a repository that retrieves data from the database

```
public class ImageRepository : IImageRepository
{
    private readonly ApplicationDbContext _context;
    public Image GetImage(Guid id)
    {
        return _context.Images.First(x => x.Id == id);
    }
}
```

```
public class ImageService : IImageService
{
    private readonly IImageRepository _imageRepository;
    public Image GetImage(Guid id)
    {
        return _imageRepository.GetImage(id);
    }
}
```

So we return the result in the controller

```
[HttpGet("Download")]
public ActionResult DownloadImage([FromQuery] Guid id)
{
    var image = _imageService.GetImage(id);
    return File(image.ImageBytes, $"image/{image.ContentType}");
}
```

## Task 1

- Create an image upload/download API with validations.
- A new functionality will be the creation of a thumbnail-type photo from the original.
- The thumbnail will need to be stored in another table.
- The point of a thumbnail is to be smaller than the original.
- You will have to look for this functionality yourself ;)
- It must be possible to get both a large photo and a thumbnail individually