Clone this project: <https://github.com/ArchwayEon/CSCI3110WorkingWithImages.git>

1. Set up the model

|  |
| --- |
| public class Image  {  public int Id { get; set; }  public string Name { get; set; } = String.Empty;  public byte[]? Data { get; set; }  public int Length { get; set; }  public string ContentType { get; set; } = String.Empty;  } |

1. Set up Entity Framework

|  |
| --- |
| PM> Install-Package Microsoft.EntityFrameworkCore  PM> Install-Package Microsoft.EntityFrameworkCore.SqlServer  PM> Install-Package Microsoft.EntityFrameworkCore.Tools |
| public class ApplicationDbContext : DbContext  {  public ApplicationDbContext(DbContextOptions options) : base(options)  {  }  public DbSet<Image> Images => Set<Image>();  } |

1. Set up the image repository. Note the use of NuGet package System.Drawing.Common.

|  |
| --- |
| public interface IImageRepository  {  Image Upload(IFormFile uploadedImage, string? name = null);  Task<Image> UploadAsync(IFormFile uploadedImage, string? name = null);  Image? Read(int id);  Task<Image?> ReadAsync(int id);  void Delete(int id);  Task DeleteAsync(int id);  } |
| public class DbImageRepository : IImageRepository  {  private readonly ApplicationDbContext \_db;  public DbImageRepository(ApplicationDbContext db)  {  \_db = db;  }  public void Delete(int id)  {  var image = Read(id);  if(image != null)  {  \_db.Images.Remove(image);  \_db.SaveChanges();  }  }  public async Task DeleteAsync(int id)  {  var image = await ReadAsync(id);  if(image != null)  {  \_db.Images.Remove(image);  await \_db.SaveChangesAsync();  }  }  public async Task<Image?> ReadAsync(int id)  {  return await \_db.Images.FirstOrDefaultAsync(i => i.Id == id);  }  public Image? Read(int id)  {  var image = \_db.Images.FirstOrDefault(i => i.Id == id);  return image;  }  public Image Upload(IFormFile uploadedImage, string? name = null)  {  MemoryStream ms = new();  uploadedImage.OpenReadStream().CopyTo(ms);  name ??= uploadedImage.Name;  Image imageEntity = new()  {  Id = 0,  Name = name,  Data = ms.ToArray(),  ContentType = uploadedImage.ContentType  };  \_db.Images.Add(imageEntity);  \_db.SaveChanges();  return imageEntity;  }  public async Task<Image> UploadAsync(IFormFile uploadedImage, string? name = null)  {  MemoryStream ms = new();  await uploadedImage.OpenReadStream().CopyToAsync(ms);  name ??= uploadedImage.Name;  Image imageEntity = new()  {  Id = 0,  Name = name,  Data = ms.ToArray(),  ContentType = uploadedImage.ContentType  };  \_db.Images.Add(imageEntity);  await \_db.SaveChangesAsync();  return imageEntity;  }  } |

1. Set up the connection string and the scoped service:

|  |
| --- |
| "ConnectionStrings": {  "DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=ImagesDb;Trusted\_Connection=True;MultipleActiveResultSets=true"  }, |
| var builder = WebApplication.CreateBuilder(args);  // Add services to the container.  builder.Services.AddControllersWithViews();  **builder.Services.AddDbContext<ApplicationDbContext>(options => {**  **options.UseSqlServer(**  **builder.Configuration.GetConnectionString("DefaultConnection"));**  **});**  **builder.Services.AddScoped<IImageRepository, DbImageRepository>();** |

1. Set up the ImageController

|  |
| --- |
| private readonly IImageRepository \_imageRepo;  public ImageController(IImageRepository imageRepo)  {  \_imageRepo = imageRepo;  }  public async Task<IActionResult> Index()  {  return View(await \_imageRepo.ReadAllAsync());  }  public async Task<IActionResult> Data(int id)  {  var image = await \_imageRepo.ReadAsync(id);  if (image == null || image.Data == null)  {  return NotFound();  }  MemoryStream ms = new MemoryStream(image.Data);  return await Task.Run(() => new FileStreamResult(ms, image.ContentType));  } |
| <table class="table">  <thead>  <tr>  <th>  @Html.DisplayNameFor(model => model.Id)  </th>  <th>  @Html.DisplayNameFor(model => model.Name)  </th>  <th>  @Html.DisplayNameFor(model => model.Length)  </th>  <th>  @Html.DisplayNameFor(model => model.ContentType)  </th>  <th>  Image  </th>  <th></th>  </tr>  </thead>  <tbody>  @foreach (var item in Model)  {  <tr>  <td>  @Html.DisplayFor(modelItem => item.Id)  </td>  <td>  @Html.DisplayFor(modelItem => item.Name)  </td>  <td>  @Html.DisplayFor(modelItem => item.Length)  </td>  <td>  @Html.DisplayFor(modelItem => item.ContentType)  </td>  <td>  <img src="/Image/Data/@item.Id" alt="@item.Name image" class="thumbnail" />  </td>  <td>  <a asp-action="Edit" asp-route-id="@item.Id">Edit</a> |  <a asp-action="Details" asp-route-id="@item.Id">Details</a> |  <a asp-action="Delete" asp-route-id="@item.Id">Delete</a>  </td>  </tr>  }  </tbody>  </table> |
| img.thumbnail {  border: 1px solid #ddd;  border-radius: 4px;  padding: 5px;  width: 150px;  } |

1. Implement Create

|  |
| --- |
| public IActionResult Create()  {  return View();  }  [HttpPost]  public async Task<IActionResult> Upload(string name, IList<IFormFile> files)  {  await TryUploadImage(files, name);  return RedirectToAction("Index");  } |
| **<form asp-action="Upload" asp-controller="Image" enctype="multipart/form-data">**  <div asp-validation-summary="ModelOnly" class="text-danger"></div>  <div class="form-group">  <label for="files" class="control-label">Image file</label>  <input id="files" name="files" type="file" class="form-control" />  @if (Model != null)  {  <img src="/Image/Details/@Model.Id" alt="@Model.Name image" width="40" height="60" />  }  <span id="filesMessages" class="text-danger"></span>  </div>  <div class="form-group">  <label asp-for="Name" class="control-label"></label>  <input asp-for="Name" class="form-control" />  <span asp-validation-for="Name" class="text-danger"></span>  </div>  <div class="form-group">  <input type="submit" value="Create" class="btn btn-primary" />  </div>  </form> |
|  |
|  |

1. Implement Details:

|  |
| --- |
| public async Task<IActionResult> Details(int id)  {  var image = await \_imageRepo.ReadAsync(id);  if (image == null || image.Data == null)  {  return RedirectToAction("Index");  }  return View(image);  } |
| <dt class="col-sm-2">  Image  </dt>  <dd class="col-sm-10">  <img src="/Image/Data/@Model.Id" alt="@Model.Name image" />  </dd> |
|  |