### Initial State

Start with a new Asp.Net Core Web Application using the MVC template with Individual User Accounts.

### Add a Photo class

1. Right-click Models - add - Class

Name Photo

1. Add properties

gg

1. Check for errors by building.

### Add a DbSet property to the DbContext class

1. Open data - ApplicationDbContext
2. Add using directive

using WebApplication.Models;

1. Add property

public DbSet<Photo> Photos { get; set; }

1. Check for errors by building.

### Migrate changes to the database

1. Tools – NuGet Package Manager – Package Manager Console
2. add-migrations newPhotoModel
3. update-database

### Create the Photos Controller

1. Right-click controllers - add - controller
2. MVC Controller with views, using Entity Framework

Model - Photo (WebApplication.Models)

Context - ApplicationDbContext (WebApplication.Data)

* 1. Add MVC Controller With views. using Entity Framework 
     Model class: 
     Qata context class: 
     Views: 
     Genera te Views 
     2hoto (WebApp"catlonModeIs) 
     licationDbContext (WebA lication_Data) 
     @ Reference script libraries 
     a layout page: 
     (Leave empty if it is set in a Razor _viewstart file) 
     Controller name: 
     PhotosController 

### Enhance controller

1. Add using directive

using Microsoft.AspNetCore.Http;

1. Add additional parameter for uploaded file.

public async Task<IActionResult> Create([Bind("PhotoID,Title,PhotoData,MimeType")] Photo photo, List<IFormFile> fileInputData)

1. Enhance code to upload file to database

if (ModelState.IsValid)

{

if (fileInputData.Count > 1)

{

ModelState.AddModelError("PhotoData", "Please upload only one file.");

}

var formFile = fileInputData[0];

var readStream = formFile.OpenReadStream();

photo.PhotoData = new byte[formFile.Length];

readStream.Read(photo.PhotoData, 0, (int)formFile.Length);

photo.MimeType = formFile.ContentType;

\_context.Add(photo);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(photo);

### Enhance create view

* Open views\photos\create
* Alter form to use multipart encoding
* <form asp-action="Create" enctype="multipart/form-data">
* Replace PhotoData and MimeType input with a single file input.
* Old

~~<div class="form-group">~~

~~<label asp-for="PhotoData" class="control-label"></label>~~

~~<input asp-for="PhotoData" class="form-control" />~~

~~<span asp-validation-for="PhotoData" class="text-danger"></span>~~

~~</div>~~

~~<div class="form-group">~~

~~<label asp-for="MimeType" class="control-label"></label>~~

~~<input asp-for="MimeType" class="form-control" />~~

~~<span asp-validation-for="MimeType" class="text-danger"></span>~~

~~</div>~~

* New

<div class="form-group">

<label name="fileInputData" class="control-label"></label>

<input name="fileInputData" class="form-control" type="file" />

<span asp-validation-for="PhotoData" class="text-danger"></span>

</div>

### Add action to return image.

* Open controllers\PhotosController.cs
* Add action to controller.

public async Task<IActionResult> File(int? id)

{

if (id == null)

{

return NotFound();

}

var photo = await \_context.Photos

.FirstOrDefaultAsync(m => m.PhotoID == id);

if (photo == null)

{

return NotFound();

}

return File(photo.PhotoData, photo.MimeType);

}

### Edit views to return image

* Open views\photos\details
* Old

~~<dt class = "col-sm-2">~~

~~@Html.DisplayNameFor(model => model.PhotoData)~~

~~</dt>~~

~~<dd class = "col-sm-10">~~

~~@Html.DisplayFor(model => model.PhotoData)~~

~~</dd>~~

~~<dt class = "col-sm-2">~~

~~@Html.DisplayNameFor(model => model.MimeType)~~

~~</dt>~~

~~<dd class = "col-sm-10">~~

~~@Html.DisplayFor(model => model.MimeType)~~

~~</dd>~~

* New

<dt class = "col-sm-2">

Photo

</dt>

<dd class = "col-sm-10">

<img src="@Url.Action("File", "Photos", new { id = Model.PhotoID })" />

</dd>

* Open views\photos\index
* Replace
* Old
* New

<td>

<img src="@Url.Action("File", "Photos", new { id = item.PhotoID })"/>

</td>