Walkthrough 18 - Razor Pages with EF Core

Setup

This lab will add database access to a Razor Pages site.

1. Start SQL Server.
2. Start Visual Studio.
3. Click Create a new project.
4. Set language to **C#** and project type to **Web**.
5. Select the ASP.NET Core Web App template, click Next.
6. Set Project name to **MovieTrackerRazor**.
7. Set Location to a **folder of your choosing**.
8. Ensure Place solution and project in the same directory is not selected, click Next.
9. Set version to **.NET 5.0**, unselect Configure for HTTPS, click Create.

Movie.cs

1. Create a **Models** folder.
2. Create a **Movie** class in the Models folder. Add the **using System.ComponentModel.DataAnnotations;** directive.
3.
```
public class Movie
{
    [Key]
    public int Id { get; set; }

    public string Title { get; set; }

    [DataType(DataType.Date), Display(Name = "Date Seen")]
    public DateTime? DateSeen { get; set; }

    public string Genre { get; set; }

    [Range(1, 10)]
    public int? Rating { get; set; }

    [Display(Name = "Image File")]
    public string ImageFile { get; set; }
}
```
4. Save the file.

Database Scaffolding

1. Under the Pages folder, create a folder named **Movies**.
2. Right-click the Movies folder and select Add / New Scaffolded Item... .
3. Select Razor Pages, then Razor Pages using Entity Framework (CRUD), click Add.
4. Set Model class to **Movie (MovieTrackerRazor.Models)**.
5. For Data context class, click the **+** button. Accept the name **MovieTrackerRazor.Data.MovieTrackerRazorContext**, click Add.
6. Click Add.
7. The Scaffolder will create a .cshtml and .cshtml.cs file for Create, Delete, Details, Edit and Index.
8. Briefly examine the files.
9. It will also register the database in Startup.cs.

appsettings.json

1. Update the connection string to use SQL Server Express and simplify the database name.
2.
```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "MovieTrackerRazorContext": "Server=~~(localdb)~~localhost\\~~mssqllocaldb~~sqlexpress;Database=~~MovieTrackerRazorContext-bcd6a572-65ea-4f92-ae33-9a67c275470a~~movie_tracker_razor;Trusted_Connection=True;MultipleActiveResultSets=true"
  }
}
```
3. Save the file.

Database Migration

1. In the Package Manager Console, issue the command **Add-Migration init**.
2. Followed by **Update-Database**.

_Layout.cshtml

1. Open Pages / Shared / _Layout.cshtml.
2. Change the Home menu entry to link to the Movie page.
3.
```
...
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-page="/Movies/Index">~~Home~~Movies</a>
</li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-page="/Privacy">Privacy</a>
</li>
...
```
4. Save the file.

SeedData.cs

1. Add a class named **SeedData** to the Models folder.
2. Make the class static and add an initializer method. Add the **using MovieTrackerRazor.Data;** directive.
3.
```
public static class SeedData
{
    public static void Initialize(MovieTrackerRazorContext context)
    {
    }
}
```
4. Have the initializer add 5 movies.
5.
```
public static void Initialize(MovieTrackerRazorContext context)
{
    context.Movie.AddRange(
        new Movie
        {
            Title = "The Shawshank Redemption",
            DateSeen = DateTime.Now.AddDays(-150).Date,
            Genre = "Drama",
            Rating = 8,
            ImageFile = "shawshank.jpg"
        },
        new Movie
        {
            Title = "Men in Black",
            DateSeen = DateTime.Now.AddDays(-250).Date,
            Genre = "Action",
            Rating = 7,
            ImageFile = "meninblack.jpg"
        },
        new Movie
        {
            Title = "The Dark Knight",
            DateSeen = DateTime.Now.AddDays(-350).Date,
            Genre = "Action",
            Rating = 9,
            ImageFile = "darkknight.jpg"
        },
        new Movie
        {
            Title = "12 Angry Men",
            DateSeen = DateTime.Now.AddDays(-450).Date,
            Genre = "Drama",
            Rating = 7,
            ImageFile = "12angrymen.jpg"
        },
        new Movie
        {
            Title = "Back to the Future",
            DateSeen = DateTime.Now.AddDays(-550).Date,
            Genre = "Adventure",
            Rating = 8,
            ImageFile = "backtofuture.jpg"
        }
    );
    context.SaveChanges();
}
```
6. Save the file.

Index.cshtml.cs

1. Update the constructor in Pages / Movies / Index.csthml.cs to seed the database if necessary.
2.
```
public IndexModel(MovieTrackerRazor.Data.MovieTrackerRazorContext context)
{
    _context = context;

    if (!_context.Movie.Any())
    {
        SeedData.Initialize(_context);
    }
}
```
3. Run the site and access the movies.
4. Add a property to IndexModel to support filtering.

5.
```
public class IndexModel : PageModel
{
    private readonly MovieTrackerRazor.Data.MovieTrackerRazorContext _context;

    public IndexModel(MovieTrackerRazor.Data.MovieTrackerRazorContext context)
    {
        _context = context;

        if (!context.Movie.Any())
        {
            SeedData.Initialize(context);
        }
    }

    public IList<Movie> Movie { get;set; }

    [BindProperty(SupportsGet = true)]
    public string SearchString { get; set; }

    public async Task OnGetAsync()
    {
        Movie = await _context.Movie.ToListAsync();
    }
}
```

6. Update OnGetAsync to filter the movies if a search string is specified.

7.
```
public async Task OnGetAsync()
{
    var movies = from m in _context.Movie
                 select m;

    if(!string.IsNullOrEmpty(SearchString))
    {
        movies = movies.Where(m => m.Title.Contains(SearchString));
    }

    Movie = await ~~_context.Movie~~movies.ToListAsync();
}
```

8. Run the site, navigate to movies and change the URL to **http://localhost:12345/Movies?searchstring=men**.

Index.csthml

1. Add a text field and button to allow filtering.
2.
```
...
<p>
    <a asp-page="Create">Create New</a>
</p>

<form>
    <p>
        Title: <input type="text" asp-for="SearchString" />
        <input type="submit" value="Filter" class="btn btn-primary" />
    </p>
</form>

<table class="table">
...
```

3. Run the site and search by title.

Index.csthml.cs

1. Add two more properties to IndexModel to support a genre drop-down list, add the **using Microsoft.AspNetCore.Mvc.Rendering;** directive.
2.
```
...
[BindProperty(SupportsGet = true)]
public string SearchString { get; set; }

public SelectList Genres { get; set; }

[BindProperty(SupportsGet = true)]
public string MovieGenre { get; set; }

public async Task OnGetAsync()
...
```

3. Update OnGetAsync to prepare a list of genres.
4.
```
public async Task OnGetAsync()
{
    // Use LINQ to get a list of genres
    IQueryable<string> genreQuery = from m in _context.Movie
                                    orderby m.Genre
                                    select m.Genre;
    Genres = new SelectList(await genreQuery.Distinct().ToListAsync());

    var movies = from m in _context.Movie
                 select m;

    if(!string.IsNullOrEmpty(SearchString))
    {
        movies = movies.Where(m => m.Title.Contains(SearchString));
    }

    if (!string.IsNullOrEmpty(MovieGenre))
    {
```

```
        movies = movies.Where(m => m.Genre == MovieGenre);
    }

    Movie = await movies.ToListAsync();
}
```

1. Save the file.

Index.cshtml

1. Add a drop-down list for genre.
2.
```
...
<form>
    <p>
        Title: <input type="text" asp-for="SearchString" />
        <input type="submit" value="Filter" class="btn btn-primary" />
        Genre: <select asp-for="MovieGenre" asp-items="Model.Genres" onchange="submit()">
                    <option value="">All</option>
               </select>
    </p>
</form>
...
```

3. Run the site and filter by title and/or Genre.

Images

1. In the wwwroot folder, create a new folder named **images**.
2. Download **MovieTrackerRazor_images.zip (https://mycanvas.mohawkcollege.ca/courses/66435/files/11573746?wrap=1)** **(https://mycanvas.mohawkcollege.ca/courses/66435/files/11573746/download?download_frd=1)** .
3. Unzip the 5 files.
4. Drag the file into the project in the wwwroot / images folder.

Index.cshtml

1. The table is going to be replaced with the **Bootstrap card system** **(https://getbootstrap.com/docs/4.0/components/card/)** to layout the movies.
2. Delete the existing table.
3.
```
...
</form>

<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Movie[0].Title)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Movie[0].DateSeen)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Movie[0].Genre)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Movie[0].Rating)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Movie[0].ImageFile)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
@foreach (var item in Model.Movie) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DateSeen)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Genre)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Rating)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.ImageFile)
            </td>
            <td>
                <a asp-page="./Edit" asp-route-id="@item.Id">Edit</a> |
                <a asp-page="./Details" asp-route-id="@item.Id">Details</a> |
                <a asp-page="./Delete" asp-route-id="@item.Id">Delete</a>
            </td>
        </tr>
}
    </tbody>
</table>
```

4. Replace the table with a card presentation.
5.
```
...
</form>
```

```
<div class="card-columns">
    @foreach (var item in Model.Movie)
    {
        <a asp-page="./Details" asp-route-id="@item.Id">
            <div class="card" style="width: 18rem;">
                <img src="~/images/@item.ImageFile" class="card-img-top" />
                <div class="card-body">
                    <h5 class="card-title text-center text-body">@item.Title</h5>
                </div>
            </div>
        </a>
    }
</div>
```

6. Run the site.

Create.cshtml.cs

1. Create a property for the host environment, add the **using Microsoft.AspNetCore.Hosting;** directive. Modify the constructor to accept the host environment and assign it.
2.
```
public class CreateModel : PageModel
{
    private readonly MovieTrackerRazor.Data.MovieTrackerRazorContext _context;
    private readonly IWebHostEnvironment _environment;

    public CreateModel(MovieTrackerRazor.Data.MovieTrackerRazorContext context, IWebHostEnvironment environment)
    {
        _context = context;
        _environment = environment;
    }

    public IActionResult OnGet()
    ...
```

3. Add an IFormFile property named **Upload**, add the **using Microsoft.AspNetCore.Http;** directive.
4.
```
...
public IActionResult OnGet()
{
    return Page();
}

[BindProperty]
public Movie Movie { get; set; }

[BindProperty]
public IFormFile Upload { get; set; }

public async Task<IActionResult> OnPostAsync()
...
```

5. Update OnPostAsync to upload the file to the images folder and to update the ImageFile with the file name. Add the **using System.IO;** directive.
6.
```
public async Task<IActionResult> OnPostAsync()
{
    if (!ModelState.IsValid)
    {
        return Page();
    }

    if (Upload != null)
    {
        var file = Path.Combine(_environment.WebRootPath, "images", Path.GetFileName(Upload.FileName));
        using (var fileStream = new FileStream(file, FileMode.Create))
        {
            await Upload.CopyToAsync(fileStream);
        }

        Movie.ImageFile = Path.GetFileName(Upload.FileName);
    }

    _context.Movie.Add(Movie);
    await _context.SaveChangesAsync();

    return RedirectToPage("./Index");
}
```

7. Save the file.

Create.cshtml

1. Modify the form tag to allow the uploading of files.
2.
```
...
<div class="col-md-4">
    <form method="post" enctype="multipart/form-data">
        <div asp-validation-summary="ModelOnly" class="text-danger"></div>
        <div class="form-group">
        ...
```

3. Change the ImageFile property to be an Upload control.
4.
```
...
<div class="form-group">
    <label asp-for="Movie.ImageFile" class="control-label"></label>
    <input asp-for="Movie.ImageFile" class="form-control" />
    <input type="file" asp-for="Upload" />
    <span asp-validation-for="Movie.ImageFile" class="text-danger"></span>
```

```
</div>
...
```

5. Search the web for a movie poster and download it.
6. Run the site. Add a movie with the downloaded poster.
7. Examine the wwwroot / images folder and notice that the new image is there.