Grigore Sandi Gabriel
gabigrigore17@gmail.com
ACE CEN anul II
Craiova
0766355186

RoWeb Test Stagiu de Practica

15 Aprilie 2021

Rezumat

În acest document urmează sa descriu în detaliu procesul de rezolvare a testului pentru a facilita evaluarea acestuia.

Codul sursa poate fi găsit aici:

https://github.com/GabrielGrigore17/RoWeb-Admission-Test/tree/main/Test_Two/MvcMovieFinal

Obiective

1. Afisarea paginata a intrărilor din baza de date.
2. Adaugarea opțiunii de a scrie recenzii pentru fiecare film.

1. Afisarea paginata a intrărilor din baza de date

Pasul 1:

Am adaugat clasa “PaginatedList” după cum este specificat în tutorial și am modificat task-ul **“Index”** din **“\Controllers\MoviesController.cs”** pentru a îi implementa functionalitatea.

```
4 references
public class PaginatedList<T> : List<T>
{
    3 references
    public int PageIndex { get; private set; }
    2 references
    public int TotalPages { get; private set; }

    1 reference
    public PaginatedList(List<T> items, int count, int pageIndex, int pageSize)
    {
        PageIndex = pageIndex;
        TotalPages = (int)Math.Ceiling(count / (double)pageSize);

        this.AddRange(items);
    }

    0 references
```

```
3 references
public async Task<IActionResult> Index(string movieGenre, string searchString, int? pageNumber)
{
    // Use LINQ to get list of genres.
    IQueryable<string> genreQuery = from m in _context.Movie
                                    orderby m.Genre
                                    select m.Genre;

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

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

    if (!string.IsNullOrEmpty(movieGenre))
    {
        movies = movies.Where(x => x.Genre == movieGenre);
    }

    int pageSize = 3;
    var movieGenreVM = new MovieGenreViewModel
    {
        Genres = new SelectList(await genreQuery.Distinct().ToListAsync()),
        //Movies = await movies.ToListAsync()
        Movies = await PaginatedList<Movie>.CreateAsync(movies.AsNoTracking(), pageNumber ?? 1, pageSize)
    };

    return View(movieGenreVM);
}
```

Pasul 2:

Am adaugat un simplu tabel în **“\Views\MoviesIndex.cshtml”** care se folosește de sintaxa c# pentru a itera prin intrările din baza de date, adaugand cate un anchor tag pentru fiecare pagina pe aceeași linie.

```
<table>
    <tr>
        @for (int i = 1; i <= Model.Movies.Count / 3 + 3; i++)
        {
        <td>
            <a asp-action="Index"
               asp-route-pageNumber="@(i)"
               class="btn btn-default">
                @i
            </a>
        </td>
        }
        <td></td>
    </tr>
</table>
```

2. Adaugarea opțiunii de a scrie recenzii pentru fiecare film.

Acest obiectiv a fost de departe cel mai complex astfel că au fost necesare multe modificări.

Pasul 1:

Evident primul pas a fost sa adaug un model pentru cum ar trebui sa arate un review, așa ca am adaugat **“\Models\Review.cs”**.

Pe langa campurile absolut necesare **(Id, Name, ReviewText, ReviewDate)** am adăugat și un camp “**MovieId**” pentru a putea tine cont în tabelul din baza de date de relația dintre filme și review-uri. Avand acest camp, putem filtra tabelul cu review-uri pentru a obține la output doar review-urile adresate filmului curent.

```
5 references
public class Review
{
    5 references
    public int Id { get; set; }

    2 references
    public int MovieId { get; set; }

    [StringLength(60, MinimumLength = 3)]
    [Required]
    3 references
    public string Name { get; set; }

    [Display(Name = "Review")]
    [StringLength(140, MinimumLength = 3)]
    [Required]
    3 references
    public string ReviewText { get; set; }

    [Display(Name = "Review Date")]
    [DataType(DataType.Date)]
    3 references
    public DateTime ReviewDate { get; set; }

}
```

Pasul 2:

Am declarat noul tabel in **“\Data\McvMovieContext”** și am executat un nou **migration** către baza de date prin **“\Migrations\{TimeStamp}_ReviewCreation.cs”**. Am generat via scaffolding și un controller fara **CRUD Views** (**“\Controllers\ReviewsController.cs”**) insa s-a dovedit a fi inutil.

```
public DbSet<Movie> Movie { get; set; }
9 references
public DbSet<Review> Review { get; set; }
```

```
migrationBuilder.CreateTable(
    name: "Review",
    columns: table => new
    {
        Id = table.Column<int>(type: "int", nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        MovieId = table.Column<int>(type: "int", nullable: false),
        Name = table.Column<string>(type: "nvarchar(60)", maxLength: 60, nullable: false),
        ReviewText = table.Column<string>(type: "nvarchar(140)", maxLength: 140, nullable: false),
        ReviewDate = table.Column<DateTime>(type: "datetime2", nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_Review", x => x.Id);
    });
```

Pasul 3:

Am creat un nou **Model** (**“\Models\BigViewModel.cs”** nume deloc inventiv) pentru a putea facilita utilizarea atât a unui model de **movie** cât și a unei **liste de review-uri** în interiorul **“\Views\Movies\Details.cshtml”.** De asemenea am adaugat cele două proprietăți “**Name**” și “**ReviewText**” pentru a trimite conținutul form-ului de review către **server**.

```
public class BigViewModel
{
    12 references
    public Movie Movie { get; set; }
    5 references
    public List<Review> Review { get; set; }

    public string Name;
    public string ReviewText;
```

Pasul 4

Am adus destul de multe modificări în task-ul “**Details**” din **“\Controllers\MoviesController.cs”**.

În primul rand mă folosesc de campurile **Name** și **ReviewText** (primite printr-un form din pagina “**\Views\Movies\Details.cshtml”**) pentru a crea o nouă intrare în tabelul de review-uri din baza de date. În acest proces, campul **ReviewDate** primește automat data de astăzi iar cheia externă

MovieId primește **Id-ul** filmului curent. Dacă totul este ok, salvăm și returnam o instanță actualizată a paginii curente (contine și review-ul pe care doar ce l-am trimis).

În al doilea rand, creăm o variabila **reviews** în care memoram toate intrările din tabelul cu review-uri din baza de date după care filtram aceste intrări după match-ul dintre **MovieId** și **Id-ul** filmului curent.

În cele din urma creem o instanță a **BigViewModel** pentru a împacheta datele și o trimitem catre pagina **Details.cshtml**.

```
// GET: Movies/Details/5
1 reference
public async Task<IActionResult> Details(int? id, string Name, string ReviewText)
{
    if (id == null)
    {
        return NotFound();
    }

    var movie = await _context.Movie
        .FirstOrDefaultAsync(m => m.Id == id);

    if (!string.IsNullOrEmpty(Name) && !string.IsNullOrEmpty(ReviewText))
    {
        var newReview = new Review
        {
            Name = Name,
            ReviewText = ReviewText,
            ReviewDate = DateTime.Today,
            MovieId = movie.Id
        };
        _context.Review.Add(newReview);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Details));
    }


    var reviews = from r in _context.Review
                  select r;

    reviews = reviews.Where(x => x.MovieId == movie.Id);
    if (movie == null)
    {
        return NotFound();
    }
    var bigViewModel = new BigViewModel
    {
        Movie = movie,
        Review = await reviews.ToListAsync()
    };
    return View(bigViewModel);
}
```

Pasul 5:

În pagina **“\Views\Movies\Details.cshtml”** aducem următoarele modificări.

Modificăm proprietatea **@model** pentru a menționa noul pachet de date (din **MovieGenreViewModel** devine **BigViewModel**) și aducem toate corecturile de sintaxa necesare pentru a păstra functionalitatea deja existentă.

Afisam un tabel cu review-uri structurat după modelul tabelului cu filme.

Adăugăm un form prin care putem extrage de la utilizator valorile pentru **Name** și **ReviewText** menționate la pasul anterior.

```
@model MvcMovie.Models.BigViewModel
```

```
<table class="table">
    <colgroup>
        <col span="1" style="width: 15%;">
        <col span="1" style="width: 70%;">
        <col span="1" style="width: 15%;">
    </colgroup>
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Review[0].Name)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Review[0].ReviewText)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Review[0].ReviewDate)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model.Review)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Name)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.ReviewText)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.ReviewDate)
                </td>
            </tr>
        }
    </tbody>
</table>
```

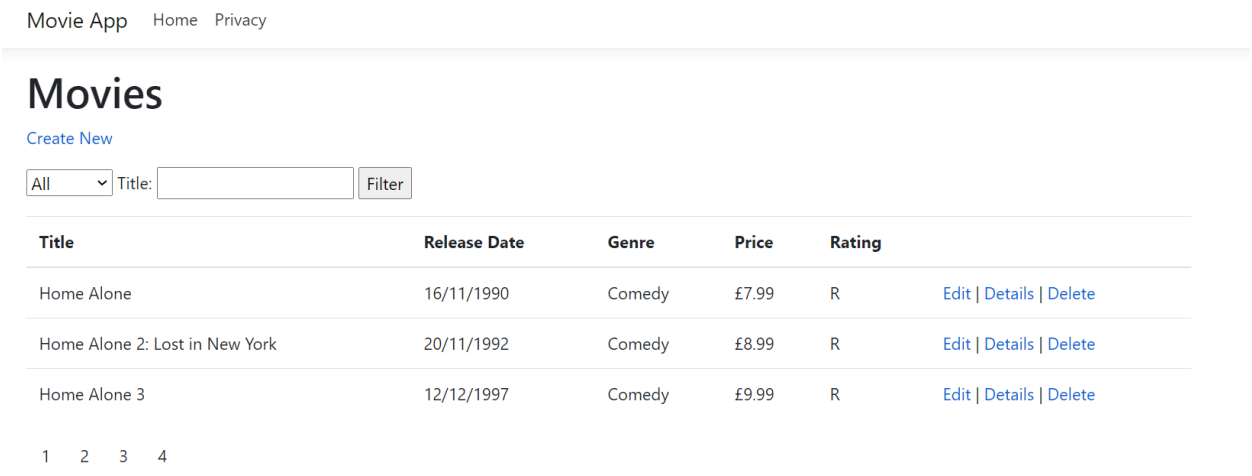
```
<div class="row">
    <div class="col-md-4">
        <form asp-controller="Movies" asp-action="Details" method="get">
            <div class="form-group">
                <label asp-for="Name" class="control-label"></label>
                <input asp-for="Name" class="form-control" type="text"/>
            </div>
            <div class="form-group">
                <label asp-for="ReviewText" class="control-label">Review</label>
                <input asp-for="ReviewText" class="form-control" type="text" />
            </div>
            <div class="form-group">
                <input type="submit" value="Send Review" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>
```

Observatii:

Deși din punct de vedere funcțional aceasta aplicatie își duce la bun sfarsit scopul, tin sa mentionez ca procesul de comunicare cu server-ul implementat de mine nu îndeplinește nici un protocol de siguranta precum HTTP, spre deosebire de procesul de adaugare al unui film (implementat după tutorial). În cazul în care aceste slăbiciuni reduc din validitatea soluției prezentate, voi reveni cu o versiune care respecta aceste protocoale de validare a informației. In caz contrar voi continua sa lucrez in ASP.NET Core pentru proiectele mele sau (sper) pentru proiectele din cadrul stagiului de practica :).

În cazul în care ai citit acest document până în acest punct, îți urez o zi minunată :)

Voi atașa în continuare cateva printscreen-uri cu aplicația.



Movie App Home Privacy

Movies

Create New

All Title: Filter

Title	Release Date	Genre	Price	Rating	
Inception	11/02/2010	Thriller	£21.00	R	Edit | Details | Delete
La vita e Bella	04/05/1985	Drama	£21.00	R	Edit | Details | Delete
Miami Bici	20/12/2020	Comedy	£20.00	R	Edit | Details | Delete

1 2 3 4

Movie App Home Privacy

Details

Movie

Title Home Alone 2: Lost in New York
Release Date 20/11/1992
Genre Comedy
Price £8.99
Rating R

Edit | Back to List

Name	Review	Review Date
Gabriel Grigore	Probably my favorite	15/04/2021
Gabriel Grigore	I was 7 when I watched it	15/04/2021
Gabriel Grigore	I watch it every Christmas	15/04/2021

Name

Gabriel Grigore

Review

This is a test

Send Review

Details

Movie

Title	Home Alone 2: Lost in New York
Release Date	20/11/1992
Genre	Comedy
Price	£8.99
Rating	R

Edit | Back to List

Name	Review	Review Date
Gabriel Grigore	Probably my favorite	15/04/2021
Gabriel Grigore	I was 7 when I watched it	15/04/2021
Gabriel Grigore	I watch it every Christmas	15/04/2021
Gabriel Grigore	This is a test	15/04/2021

Name

Review

Send Review