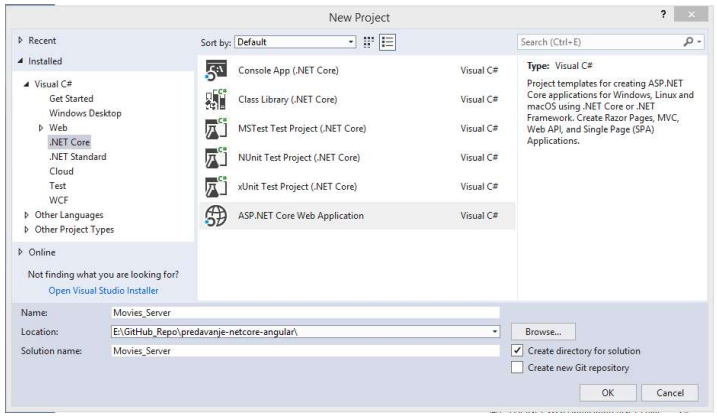
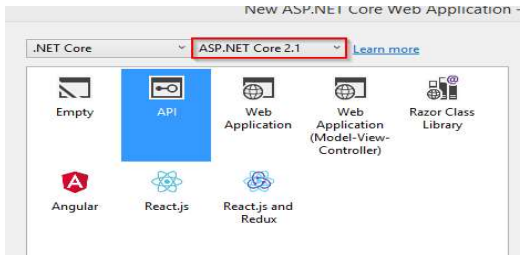
Razvoj Web Aplikacije pomoću .NET Core Web API i Angular Tehnologija

Kreiranje i Konfiguracija Projekta

.NET Core 2.1 Web API projekat





Kreiranje i Konfiguracija Projekta

Modifikacija launchSettings.json fajla.

```
{
  "$schema": "http://json.schemastore.org/launchsettings.json",
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:49899",
      "sslPort": 44322
    }
  },
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": false,
      "launchUrl": "api/values",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "Movies_Server": {
      "commandName": "Project",
      "launchBrowser": false,
      "launchUrl": "api/values",
      "applicationUrl": "https://localhost:5001;http://localhost:5000",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

app.UseHttpsRedirection();

Kreiranje i Konfiguracija Projekta

Server će slušati na portu 5001, klijent će slušati na 4200 (podrazumevano)

Zbog navedenog, mora se konfigurisati CORS na serverskoj strani projekta

Modifikacija Startup.cs klase sa **ConfigureServices** i **Configure** metodama

ConfigureServices metoda nam služi za registrovanje servisa u okviru IServiceCollection interfejsa, odnosno registrovanje servisa u IOC kontejneru

Configure metoda nam služi za dodavanje različitih middleware komponenti u okviru „pipeline“-a aplikacije.

Kreiranje i Konfiguracija Projekta

```
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddCors(options =>
        {
            options.AddPolicy("CorsPolicy",
                builder => builder.AllowAnyOrigin()
                .AllowAnyMethod()
                .AllowAnyHeader()
                .AllowCredentials());
        });

        services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
    }

    // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
    public void Configure(IApplicationBuilder app, IHostingEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseHsts();
        }

        app.UseHttpsRedirection();
        app.UseCors("CorsPolicy");
        app.UseMvc();
    }
}
```

Kreiranje i Konfiguracija Projekta

„Čistiji“ način za registrovanje servisa u IOC je korišćenjem **Ekstenzionih metoda (Extension Methods)**

```
app.UseHttpsRedirection();
app.UseCors("CorsPolicy");
app.UseMvc();
```

```
namespace Microsoft.AspNetCore.Builder
{
    public static class CorsMiddlewareExtensions
    {
        public static IApplicationBuilder UseCors(this IApplicationBuilder app);
        public static IApplicationBuilder UseCors(this IApplicationBuilder app, string policyName);
        public static IApplicationBuilder UseCors(this IApplicationBuilder app, Action<CorsPolicyBuilder> configurePolicy);
    }
}
```

Konfiguracija EF Core, Code First

Kreiranje „Movie“ modela

```
namespace Movies_Server.Models
{
    public class Movie
    {
        [Key]
        public Guid Id { get; set; }
        public string Name { get; set; }
        public string Genre { get; set; }
        public string Director { get; set; }
    }
}
```

Kreiranje Context klase

```
public class MovieContext: DbContext
{
    public MovieContext(DbContextOptions options)
        :base(options)
    {
    }

    public DbSet<Movie> Movies { get; set; }
}
```

Konfiguracija EF Core, Code First

Modifikacija appsettings.json fajla

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "ConnectionStrings": {
    "sqlConString": "Server=.;Database=MovieExampleDB;Trusted_Connection=True;"
  },
  "AllowedHosts": "*"
}
```

Registracija klase u **ConfigureServices** metodi u **Startup** klasi

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors(options =>
    {
        options.AddPolicy("CorsPolicy",
            builder => builder.AllowAnyOrigin()
            .AllowAnyMethod()
            .AllowAnyHeader()
            .AllowCredentials());
    });

    services.AddDbContext<MovieContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("sqlConString")));

    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
}
```

Konfiguracija EF Core, Code First

Popunjavanje inicijalnih podataka u MovieContext klasi

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Movie>().HasData(
        new Movie
        {
            Id = Guid.NewGuid(),
            Name = "Alien: Covenant",
            Genre = "Sci-Fi, Thriller",
            Director = "Ridley Scott"

        },
        new Movie
        {
            Id = Guid.NewGuid(),
            Name = "The Lord Of The Rings",
            Genre = "Adventure, Drama, Fantasy",
            Director = "Peter Jackson"
        });
}
```

Konfiguracija EF Core, Code First

Dodavanje migracije u Package Manager Console prozoru

```
PM> Add-Migration Initial_Database_Migration
Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 2.1.1-rtm-30846 initialized 'MovieContext' using provider 'Microso
To undo this action, use Remove-Migration.
PM> Update-Database
Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 2.1.1-rtm-30846 initialized 'MovieContext' using provider 'Microso
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (212ms) [Parameters=[], CommandType='Text', CommandTimeout='60']
      CREATE DATABASE [MovieExampleDB];
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (73ms) [Parameters=[], CommandType='Text', CommandTimeout='60']
      IF SERVERPROPERTY('EngineEdition') <> 5
      BEGIN
          ALTER DATABASE [MovieExampleDB] SET READ_COMMITTED_SNAPSHOT ON;
      END;
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (14ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      CREATE TABLE [__EFMigrationsHistory] (
          [MigrationId] nvarchar(150) NOT NULL,
          [ProductVersion] nvarchar(32) NOT NULL,
          CONSTRAINT [PK___EFMigrationsHistory] PRIMARY KEY ([MigrationId])
      );
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (14ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT OBJECT_ID(N'[__EFMigrationsHistory]');
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (1ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT [MigrationId], [ProductVersion]
      FROM [__EFMigrationsHistory]
      ORDER BY [MigrationId];
Microsoft.EntityFrameworkCore.Migrations[20402]
      Applying migration '20181126200554_Initial_Database_Migration'.
Applying migration '20181126200554_Initial_Database_Migration'.
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (3ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      CREATE TABLE [Movies] (
          [Id] uniqueidentifier NOT NULL,
          [Name] nvarchar(max) NULL,
          [Genre] nvarchar(max) NULL,
```

Konfiguracija EF Core, Code First

Postoji i druga opcija ukoliko je baza već napravljena a to je Db First pristup

Može se podeliti u dva dela:
1. Db First Standard gde manuelno pravimo klase koje su reprezentacija naših tabela u bazi.

Prednost je što su modeli čistiji i što imate veću kontrolu nad vašim klasama

2. Db First Reverse Engineering gde mi u suštini pomoću PM-a i komande :

```
Scaffold-DbContext "Server=.;Database=name;Trusted_Connection=True;"
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
```

generišemo naše modele automatski.

Brže je generisanje ali nemate kontrolu nad modelima i context klasom.

Konfiguracija EF Core, Code First

Bolji nacin za dodavanje inicijalnih podataka

```
public class MovieContext : DbContext
{
    public MovieContext(DbContextOptions options)
        : base(options)
    {
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.ApplyConfiguration(new MovieConfig());
    }

    public DbSet<Movie> Movies { get; set; }
}

//NE ZELIMO da imamo dve klase u istom fajlu, ovo je samo primera radi
public class MovieConfig : IEntityTypeConfiguration<Movie>
{
    public void Configure(EntityTypeBuilder<Movie> builder)
    {
        builder.HasData(
            new Movie
            {
                //properties...
            },
            new Movie
            {
                //properties...
            }
        );
    }
}
```

Kreiranje i Registracija Repository layer-a

Uvek koristite neki nivo Data layer-a. Vaš kontroler ne treba da zna išta o Context klasi niti o načinu prikupljanja podataka iz baze.

Manje je bitno da li koristite Repository Pattern ili neki drugi , ili čak i samo novi fajl u koji ćete smestiti vašu logiku prikupljanja podataka iz baze.

Mnogo je bitnije da se razdele obaveze različitih elemenata vašeg projekta.

Kreiranje i Registracija Repository layer-a

MovieRepository.cs klasa

```
public class MovieRepository: IMovieRepository
{
    private readonly MovieContext _context;

    public MovieRepository(MovieContext context)
    {
        _context = context;
    }

    public List<Movie> GetMovies() => _context.Movies.ToList();

    public Movie GetMovieById(Guid id) => _context.Movies.SingleOrDefault(m => m.Id.Equals(id));

    public void SaveMovie(Movie movie)
    {
        movie.Id = Guid.NewGuid();
        _context.Add(movie);
        _context.SaveChanges();
    }

    public void UpdateMovie(Movie dbMovie, Movie movie)
    {
        dbMovie.Map(movie);
        _context.Update(dbMovie);
        _context.SaveChanges();
    }

    public void DeleteMovie(Movie movie)
    {
        _context.Remove(movie);
        _context.SaveChanges();
    }
}
```

Kreiranje i Registracija Repository layer-a

IMovieRepository interface

```
public interface IMovieRepository
{
    List<Movie> GetMovies();
    Movie GetMovieById(Guid id);
    void SaveMovie(Movie movie);
    void UpdateMovie(Movie dbMovie, Movie movie);
    void DeleteMovie(Movie movie);
}
```

Registracija IMovieRepository servisa

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors(options =>...);

    services.AddDbContext<MovieContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("sqlConString")));

    services.AddScoped<IMovieRepository, MovieRepository>();

    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
}
```

Kreiranje Kontrolera i Akcija

Dodavanje novog MovieController kontrolera i Atribute routing

```
[Route("api/movie")]
[ApiController]
public class MovieController : ControllerBase
{
}
```

Attribute routing je primenjen i promenjen je sa api/[controller] na api/movie

Pored Attribute routing postoji i Conventional routing i on je vise primenljiv u MVC projektima

```
app.UseMvc(routes =>
{
    routes.MapRoute("default", "{controller=Home}/{action=Index}/{id?}");
});
```

Second part

Third part

First part

Kreiranje Kontrolera i Akcija

GetMovies Akcija, IActionResult i Metode za objedinjavanje povratnog tipa i Status koda

```
[Route("api/movie")]
[ApiController]
public class MovieController : ControllerBase
{
    private readonly IMovieRepository _movieRepo;

    public MovieController(IMovieRepository movieRepo)
    {
        _movieRepo = movieRepo;
    }

    [HttpGet]
    public IActionResult GetMovies()
    {
        try
        {
            var movies = _movieRepo.GetMovies();
            return Ok(movies);
        }
        catch (Exception ex)
        {
            //logovanje greske
            return StatusCode(500, $"Internal server error: {ex}");
        }
    }
}
```

Kreiranje Kontrolera i Akcija

Rezultat pomoću Postman alata

GET https://localhost:5001/api/movie Send

Params Authorization Headers Body Pre-request Script Tests

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (4) Test Results Status: 200 OK Time: 36 ms Size: 678 B

Pretty Raw Preview JSON


```
[
    {
        "id": "12366b4e-3710-4851-9275-37a533710d2e",
        "name": "Alien: Covenant",
        "genre": "Sci-Fi, Thriller",
        "director": "Ridley Scott"
    },
    {
        "id": "83a879ce-d6e6-4ff4-8a47-496226ae51bd",
        "name": "The Lord Of The Rings",
        "genre": "Adventure, Drama, Fantasy",
        "director": "Peter Jackson"
    },
    {...},
    {...}
]
```

Kreiranje Kontrolera i Akcija

GetMovie Akcija

```
[HttpGet("{id}", Name = "MovieById")]
public IActionResult GetMovie(Guid id)
{
    try
    {
        var movie = _movieRepo.GetMovieById(id);
        if(movie == null)
        {
            return NotFound();
        }

        return Ok(movie);
    }
    catch (Exception ex)
    {
        //logovanje greske
        return StatusCode(500, $"Internal server error: {ex}");
    }
}
```

Kreiranje Kontrolera i Akcija

POST Akcija

```
[HttpPost]
public IActionResult Post([FromBody]Movie movie)
{
    try
    {
        if (movie == null)
        {
            return BadRequest("Movie object is null");
        }

        if (!ModelState.IsValid)
        {
            return BadRequest("Invalid model object");
        }

        _movieRepo.SaveMovie(movie);

        return CreatedAtRoute("MovieById", new { id = movie.Id }, movie);
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"Internal server error: {ex}");
    }
}
```

Kreiranje Kontrolera i Akcija

Rezultat POST request-a

POST https://localhost:5001/api/movie

Params Authorization Headers (1) Body Pre-request Sc

none form-data x-www-form-urlencoded raw bin

```
{
    "name": "The Lord Of The Rings333",
    "genre": "Adventure, Drama, Fantasy33",
    "director": "Peter Jackson33"
}
```

Location → https://localhost:5001/api/movie/4266bfc0-e7e0-4a28-b63d-f06c72d8698c

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON

```
{
    "id": "4266bfc0-e7e0-4a28-b63d-f06c72d8698c",
    "name": "The Lord Of The Rings333",
    "genre": "Adventure, Drama, Fantasy33",
    "director": "Peter Jackson33"
}
```

Kreiranje Kontrolera i Akcija

Map extension method

```
public static class MovieExtensions
{
    public static void Map(this Movie dbMovie, Movie movie)
    {
        dbMovie.Name = movie.Name;
        dbMovie.Genre = movie.Genre;
        dbMovie.Director = movie.Director;
    }
}
```

Korišćenje u MoveRepository fajlu pomoću sintakse: dbMovie.Map(movie);

Kreiranje Kontrolera i Akcija

PUT Akcija

```
[HttpPut("{id}")]
public IActionResult Put(Guid id, [FromBody]Movie movie)
{
    try
    {
        if (movie == null)
        {
            return BadRequest("Movie object is null");
        }

        if (!ModelState.IsValid)
        {
            return BadRequest("Invalid model object");
        }

        var dbMovie = _movieRepo.GetMovieById(id);
        if (dbMovie == null)
        {
            return NotFound();
        }

        _movieRepo.UpdateMovie(dbMovie, movie);

        return NoContent();
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"Internal server error: {ex}");
    }
}
```

Kreiranje Kontrolera i Akcija

Delete Akcija

```
[HttpDelete("{id}")]
public IActionResult Delete(Guid id)
{
    try
    {
        var movie = _movieRepo.GetMovieById(id);
        if (movie == null)
        {
            return NotFound();
        }

        _movieRepo.DeleteMovie(movie);

        return NoContent();
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"Internal server error: {ex}");
    }
}
```

Dodatni Materijal za .NET Core

.Net Core Web Api Detaljnije => https://code-maze.com/net-core-series/

Global Error Handling => https://code-maze.com/global-error-handling-aspnetcore/

Action Filters kako pisati cistije akcije => https://code-maze.com/action-filters-aspnetcore/

Code First EF Core => https://code-maze.com/net-core-web-api-ef-core-code-first/

Db First EF Core => https://code-maze.com/netcore-web-api-db-first/

Best Practices .NET Core Web API => https://code-maze.com/aspnetcore-webapi-best-practices/

Content Negotiation .NET Core => https://code-maze.com/content-negotiation-dotnet-core/

Ove i mnoge druge informacije => https://www.google.rs

Priprema Angular Projekta

Pre kreiranja projekta, uvek je preporuka da instalirate Angular CLI

Olakšava kreiranje projekta, pravljenje fajlova u samom projektu, priprema fajlova za produkciju, kao i testiranje projekta.

https://github.com/angular/angular-cli/blob/master/packages/angular/cli/README.md

Zahtevan je Node.js koji u sebi ima i npm instalaciju obezbeđenu

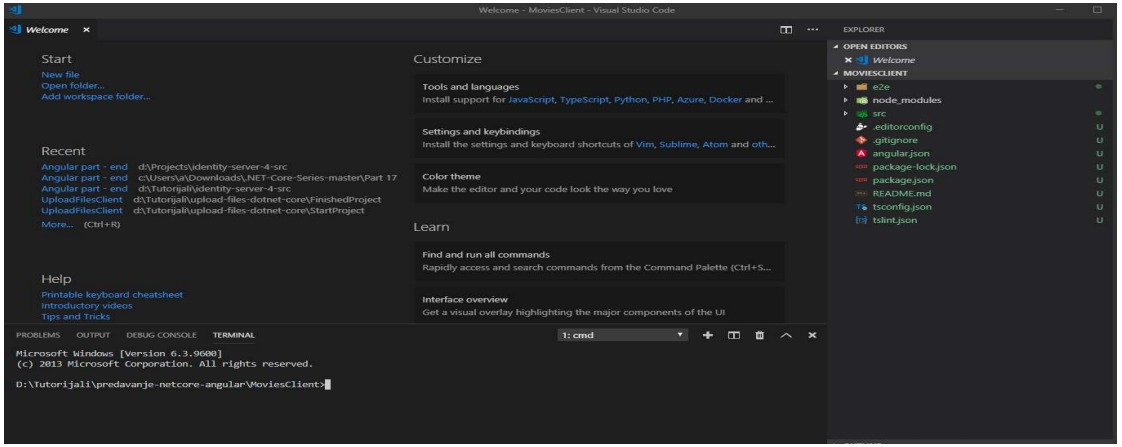
Code editor, moja preporuka Visual Studio Code

Priprema Angular Projekta

ng new MoviesClient

```
D:\Tutorijali\predavanje-netcore-angular>ng new MoviesClient
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SCSS    [ http://sass-lang.com   ]
  SASS    [ http://sass-lang.com   ]
  LESS    [ http://lesscss.org     ]
  Stylus  [ http://stylus-lang.com ]
```

Projekat otvorimo u VS Code i omogućimo Terminal Window (CTRL+`)



Priprema Angular Projekta

Instalacija Bootstrap 3 biblioteke za potrebe silizovanja aplikacije

```
D:\Tutorijali\predavanje-netcore-angular\MoviesClient>npm install --save bootstrap@3
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: want

+ bootstrap@3.3.7
added 1 package in 9.737s
```

Referenca ka node paketu unutar angular.json fajla

```
],
"styles": [
  "./node_modules/bootstrap/dist/css/bootstrap.min.css",
  "src/styles.css"
],
"scripts": []
```

Priprema Angular Projekta

Instalacija @types/bootstrap paketa komandom: npm install --save @types/bootstrap

Registracija @types unutar tsconfig.app.json fajla

Sve isto i za JQuery biblioteku.

```
"types": [
  "jquery",
  "bootstrap"
]
```

```
  "styles": [
    "./node_modules/bootstrap/dist/css/bootstrap.min.css",
    "src/styles.css"
  ],
  "scripts": [
    "./node_modules/jquery/dist/jquery.min.js",
    "./node_modules/bootstrap/dist/js/bootstrap.min.js"
  ]
```

Priprema Angular Projekta

Angular aplikacije su SPA (single page application) aplikacije zato što se serviraju u samo jednoj stranici index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MoviesClient</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Priprema Angular Projekta

Svaka komponenta se sastoji iz minimalno tri fajla: .component, .html, .css
Postoji i četvrti .spec.ts file, koji služi za testiranje...

angular.json | app.component.ts × | tsconfig.app.json

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'MoviesClient';
10 }
11
```

app
- app.component.css
- app.component.html
- app.component.spec.ts
- app.component.ts

Priprema Angular Projekta

Komponente nemaju nikakvu funkcionalnost bez modul fajlova.

Podrazumevano kreiran app.module.ts file (App Module file)

angular.json | app.component.ts | app.module.ts × | tsconfig.app.j

```
1   import { BrowserModule } from '@angular/platform-browser';
2   import { NgModule } from '@angular/core';
3
4   import { AppComponent } from './app.component';
5
6   @NgModule({
7     declarations: [
8       AppComponent
9     ],
10    imports: [
11      BrowserModule
12    ],
13    providers: [],
14    bootstrap: [AppComponent]
15  })
16  export class AppModule { }
```

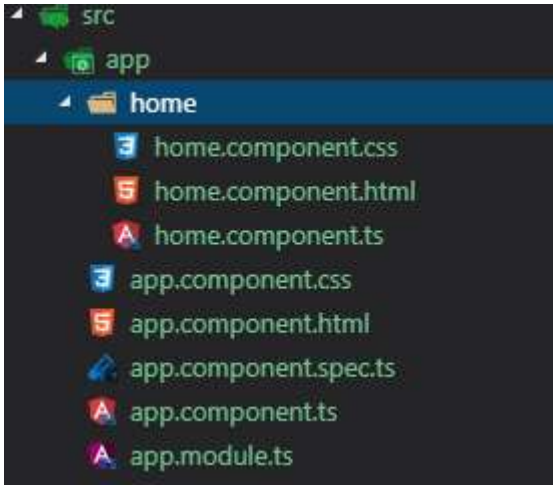
Prva Komponenta

Home komponenta prvo, koja će biti deo Application Module-a

```
D:\Tutorijali\predavanje-netcore-angular\MoviesClient>ng g component home --spec false
CREATE src/app/home/home.component.html (23 bytes)
CREATE src/app/home/home.component.ts (261 bytes)
CREATE src/app/home/home.component.css (0 bytes)
UPDATE src/app/app.module.ts (388 bytes)
```

angular.json | app.component.ts | app.module.ts

```
1  import { BrowserModule } from '@angular/platfor
2  import { NgModule } from '@angular/core';
3
4  import { AppComponent } from './app.component'
5  import { HomeComponent } from './home/home.comp
6
7  @NgModule({
8    declarations: [
9      AppComponent,
10     HomeComponent
11   ],
12   imports: [
13     BrowserModule
14   ],
15   providers: [],
16   bootstrap: [AppComponent]
17 })
18 export class AppModule { }
19
```



Prva Komponenta

Modifikujemo home.component.ts fajl prvo

gular.json | app.component.ts | home.component.ts ×

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  public homeText: string;

  constructor() { }

  ngOnInit() {
    this.homeText = "Home Component for our project.";
  }

}
```

Prva Komponenta

public modifikator pristupa je preporučljiv kada se property ili funkcija koriste u template (.html) fajlu.

OnInit je interfejs koji obavezuje implementaciju **ngOnInit** funkcije. Ova f-ja će se okinuti čim se komponenta inicijalizuje. Drugi u fazi životnog ciklusa Angular app.

Uvek koristite **ngOnInit** funkciju da bi ste inicijalizovali sve potrebne elemente u vašim komponentama.

Ne mora se koristiti OnInit interfejs da bi se koristila ngOnInit f-ja, ali je preporučljivo.

Konstruktor samo za inicijalizaciju servisa.

https://angular.io/guide/lifecycle-hooks

Prva Komponenta

Modifikacija .html i .css fajla

home.component.ts | home.component.css | home.component.html

```
1   <div class="col-md-12">
2     <p class="homeText">{{homeText}}</p>
3   </div>
4
```

home.component.ts | home.component.css

```
1   .homeText{
2       font-size: 35px;
3       color: red;
4       text-align: center;
5       position: relative;
6       top:30px;
7       text-shadow: 2px 2px 2px gray;
8   }
```

{{}} => interpolacija. Projektovanje vrednosti propertia unutar template-a.

Prva Komponenta

Modifikacija app.component.html fajla i **ng serve** komanda za startovanjej projekta

app.component.html × home.component.ts

```
1  <div class="container container-fluid">
2      <div class="row">
3          <app-home></app-home>
4      </div>
5  </div>
6
7
```

Navigacija i Routing

Kreiramo Menu komponentu sa Angular CLI

```
D:\Tutorijali\predavanje-netcore-angular\MoviesClient>ng g c menu --spec false
CREATE src/app/menu/menu.component.html (23 bytes)
CREATE src/app/menu/menu.component.ts (261 bytes)
CREATE src/app/menu/menu.component.css (0 bytes)
UPDATE src/app/app.module.ts (462 bytes)
```

Kreiramo Menu komponentu sa Angular CLI

menu.component.html

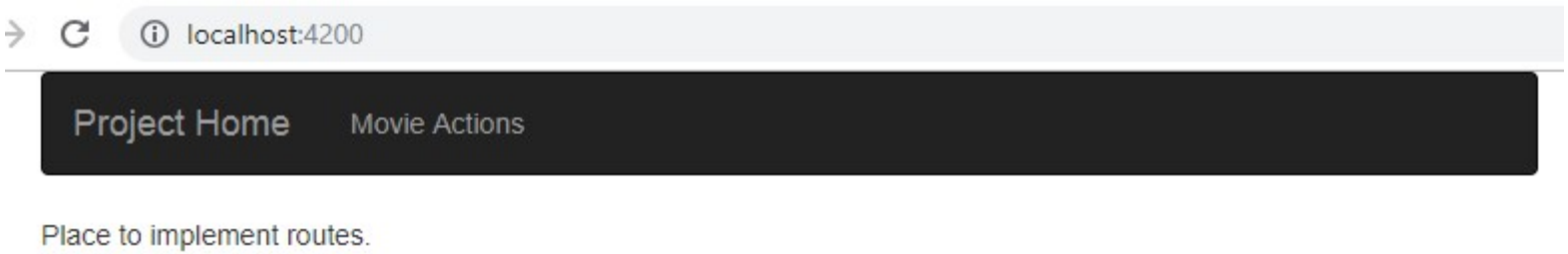
```
<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">Project Home</a>
    </div>
    <ul class="nav navbar-nav">
      <li><a href="#">Movie Actions</a></li>
    </ul>
  </div>
</nav>
```

Navigacija i Routing

Umetnuti app-menu slektor u app.component.html file

menu.component.html | app.component.html ×

```
1  <div class="container container-fluid">
2      <div class="row">
3          <app-menu></app-menu>
4      </div>
5      <div class="row">
6          Place to implement routes.
7      </div>
8  </div>
9
10
```



Navigacija i Routing

Da bi smo implementirali Routing u projektu, prvo moramo da kreiramo Routing module

```
D:\Tutorijali\predavanje-netcore-angular\MoviesClient>ng g module routing --spec false --module app
CREATE src/app/routing/routing.module.ts (191 bytes)
UPDATE src/app/app.module.ts (539 bytes)
```

Komanda će kreirati novi routing.module.ts file i registrovati ga u app.module.ts fajlu

Kada izmenimo routing file, on treba da izgleda ovako:

Navigacija i Routing

routing.module.ts × menu.component.html app.component.html

```
1   import { NgModule } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3   import { HomeComponent } from '../home/home.component';
4   import { Routes, RouterModule } from '@angular/router';
5
6   const routes: Routes = [
7     { path: 'home', component: HomeComponent},
8     { path: '', redirectTo: '/home', pathMatch: 'full' }
9
10  ];
11
12  @NgModule({
13    declarations: [],
14    imports: [
15      CommonModule,
16      RouterModule.forRoot(routes)
17    ],
18    exports: [
19      RouterModule
20    ]
21  })
22  export class RoutingModule { }
23
```

Navigacija i Routing

Sada moramo izmeniti app.component.html file

```
<div class="container container-fluid">
    <div class="row">
        <app-menu></app-menu>
    </div>
    <div class="row">
        <router-outlet></router-outlet>
    </div>
</div>
```

<router-outlet> predstavlja placeholder koji će prihvatati sadržaj komponenti koje želimo da izrenderujemo na stranici.

Navigacija i Routing

Da bi nam menu radio, moramo mu podesiti definisane rute

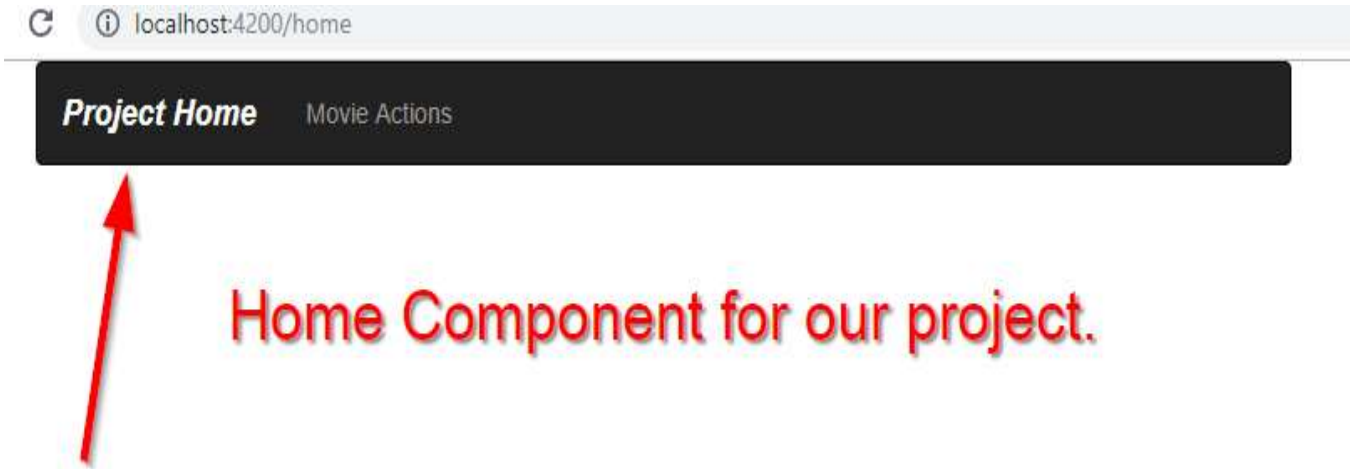
menu.component.html

```
<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" [routerLink]="['/home']" routerLinkActive="active"
        [routerLinkActiveOptions]="{exact: true}">Project Home</a>
    </div>
    <ul class="nav navbar-nav">
      <li><a href="#">Movie Actions</a></li>
    </ul>
  </div>
</nav>
```

menu.component.css

```
.active{
    font-weight: bold;
    font-style: italic;
    color: #fff;
}
```

Navigacija i Routing



“Not Found” Komponenta

Kreiranje komponente

```
D:\Tutorijali\predavanje-netcore-angular\MoviesClient>ng g c notFound --spec false
CREATE src/app/not-found/not-found.component.html (28 bytes)
CREATE src/app/not-found/not-found.component.ts (280 bytes)
CREATE src/app/not-found/not-found.component.css (0 bytes)
UPDATE src/app/app.module.ts (631 bytes)
```

Modifikacija fajlova

routing.module.ts | not-found.component.ts ×

```ts
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-not-found',
  templateUrl: './not-found.component.html',
  styleUrls: ['./not-found.component.css']
})
export class NotFoundComponent implements OnInit {

  public notFoundText: string;

  constructor() { }

  ngOnInit() {
    this.notFoundText = `404, We Couldn't find your page`;
  }

}
```

not-found.component.html ×

```html
<p>
  {{notFoundText}}
</p>
```

not-found.component.css ×

```css
p{
    font-weight: bold;
    font-size: 50px;
    text-align: center;
    color: #f10b0b;
}
```

“Not Found” Komponenta

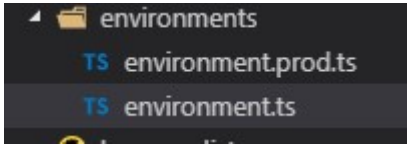
Podešavanje router moudle fajla

```
import { NotFoundComponent } from '../not-found/not-found.component';

const routes: Routes = [
  { path: 'home', component: HomeComponent},
  { path: '404', component : NotFoundComponent},
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path: '**', redirectTo: '/404', pathMatch: 'full'}
```

Home Component for our project.

Servisi i Environment Fajlovi

Environment fajlovi nam omogućavaju da prepoznamo u kom okruženju nam se nalazi aplikacija i u odnosu na to da podesimo određena podešavanja, kao npr end-point adresu.




```
export const environment = {
  production: false
};
```

```
export const environment = {
  production: true
};
```

Izmenićemo oba fajla sa odgovarajućim serverskim adresama

```
export const environment = {
  production: false,
  serverAddress: 'https://localhost:5001'
};
```

```
export const environment = {
  production: true,
  serverAddress: 'http://www.MoviesProject.com'
};
```

Posle podešavanja fajlova za okruženje, kreiraćemo servis za HTTP zahteve ka serveru.

Servisi i Environment Fajlovi

Da bi smo slali HTTP zahteve ka serveru, mi moramo da registrujemo HTTPClientModule u Application Module fajlu

app.module.ts | not-found.component.css | not-found.component.html

```
1   import { BrowserModule } from '@angular/platform-browser';
2   import { NgModule } from '@angular/core';
3   import { HttpClientModule } from '@angular/common/http';
4
5   import { AppComponent } from './app.component';
6   import { HomeComponent } from './home/home.component';
7   import { MenuComponent } from './menu/menu.component';
8   import { RoutingModule } from './routing/routing.module';
9   import { NotFoundComponent } from './not-found/not-found.component';
10
11  @NgModule({
12    declarations: [
13      AppComponent,
14      HomeComponent,
15      MenuComponent,
16      NotFoundComponent
17    ],
18    imports: [
19      BrowserModule,
20      RoutingModule,
21      HttpClientModule
22    ],
23    providers: [],
24    bootstrap: [AppComponent]
25  })
26  export class AppModule { }
```

Servisi i Environment Fajlovi

Sada možemo napraviti naš httpService fajl

```
E:\GitHub_Repo\predavanje-netcore-angular\MoviesClient>ng g service shared\http --spec false
CREATE src/app/shared/http.service.ts (133 bytes)
```

Podrazumevano naš servis izgleda ovako

http.service.ts

```
1  import { Injectable } from '@angular/core';
2
3  @Injectable({
4    providedIn: 'root'
5  })
6  export class HttpService {
7
8    constructor() { }
9  }
```

Servisi su fajlovi u koje možemo da izvučemo logku relavantnu za naše komponente isto tako logiku koju možemo ponovno koristiti u drugim komponentama.

http.service.ts

```typescript
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { environment } from './../../environments/environment';

@Injectable({
  providedIn: 'root'
})
export class HttpService {

  constructor(private http: HttpClient) { }

  public getData = (route: string) => { return this.http.get(this.createCompleteRoute(route, environment.serverAddress)) };

  public create = (route: string, body) =>  { return this.http.post(this.createCompleteRoute(route, environment.serverAddress), body, this.generateHeaders()) };

  public update = (route: string, body) => { return this.http.put(this.createCompleteRoute(route, environment.serverAddress), body, this.generateHeaders()) };

  public delete = (route: string) => { return this.http.delete(this.createCompleteRoute(route, environment.serverAddress)) };

  private createCompleteRoute = (route: string, envAddress: string) => `${envAddress}/${route}`;

  private generateHeaders = () => {
    return {
      headers: new HttpHeaders({'Content-Type': 'application/json'})
    }
  }
}
```

Servisi i Environment Fajlovi

Pre Arrow Funkcije

```
public getData(route: string){
  return this.http.get(this.createCompleteRoute(route, environment.serverAddress));
}
```

Arrow Funkcija implementirana

```
public getData = (route: string) => {
  return this.http.get(this.createCompleteRoute(route, environment.serverAddress));
}
```

Servisi i Environment Fajlovi

Rezultat http akcija je uvek Observable<…>

```
(method) HttpClient.get(url: string, options?: {
    headers?: HttpHeaders | {
        [header: string]: string | string[];
    };
    observe?: "body";
    params?: HttpParams | {
        [param: string]: string | string[];
    };
    reportProgress?: boolean;
    responseType?: "json";
    withCredentials?: boolean;
}): Observable<...> (+14 overloads)
```

```
ttp.get(this.createCompleteRoute(route, environment.serverAddress));
```

Observable je asinhrone prirode i zbog toga ga uvek moramo sačekati ako želimo da iskoristimo povratne podatke.

Koristimo **Subscribe** da bi smo sačekali response od Observable

Lazy Loading i Prikaz Podataka

Lazy loading je feature koji omogućava da se resursi aplikacije učitavaju samo onda kada su zaista i potrebni.

Ubrzava rad i učitavanje same aplikacije

Kreiranjem dodatnih modula u aplikaciji formiramo okruženje za korišćenje Layz Loading feature-a

Kreiraćemo novi Movie modul fajl

```
E:\GitHub_Repo\predavanje-netcore-angular\MoviesClient>ng g module movie
CREATE src/app/movie/movie.module.ts (189 bytes)
```

movie.module.ts | http-servic.service.ts | app.module.ts

```
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3
4  @NgModule({
5    declarations: [],
6    imports: [
7      CommonModule
8    ]
9  })
10 export class MovieModule { }
11
```

Lazy Loading i Prikaz Podataka

Treba da izmenimo app.module routing fajl, da bi smo učitali naš novi module kroz Lazy Load feature

```
const routes: Routes = [
  { path: 'home', component: HomeComponent},
  { path: 'movie', loadChildren: "../movie/movie.module#MovieModule" },
  { path: '404', component : NotFoundComponent},
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path: '**', redirectTo: '/404', pathMatch: 'full'}

];
```

Napravićemo novu komponentu Movies, za prikaz svih filmova iz baze

```
E:\GitHub_Repo\predavanje-netcore-angular\MoviesClient>ng g c movie/movies --spec false
CREATE src/app/movie/movies/movies.component.html (25 bytes)
CREATE src/app/movie/movies/movies.component.ts (269 bytes)
CREATE src/app/movie/movies/movies.component.css (0 bytes)
UPDATE src/app/movie/movie.module.ts (273 bytes)
```

Lazy Loading i Prikaz Podataka

Sada hoćemo da napravimo novi routing file za naš novi Movie modul

```
E:\GitHub_Repo\predavanje-netcore-angular\MoviesClient>ng g module movie/movie-routing --module movie
CREATE src/app/movie/movie-routing/movie-routing.module.ts (196 bytes)
UPDATE src/app/movie/movie.module.ts (372 bytes)
```

I izmeniti ga da učitamo našu Movies komponentu

movie.module.ts | movie-routing.module.ts × | routing.module.ts

```
1   import { NgModule } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3   import { MoviesComponent } from '../movies/movies.component';
4   import { Routes, RouterModule } from '@angular/router';
5
6   const routes: Routes = [
7     { path: 'list', component: MoviesComponent }
8   ];
9
10  @NgModule({
11    declarations: [],
12    imports: [
13      CommonModule,
14      RouterModule.forChild(routes)
15    ],
16    exports: [
17      RouterModule
18    ]
19  })
20  export class MovieRoutingModule { }
```

Lazy Loading i Prikaz Podataka

Izmenićemo i menu.component.html fajl, da učitamo rutu za novu komponentu

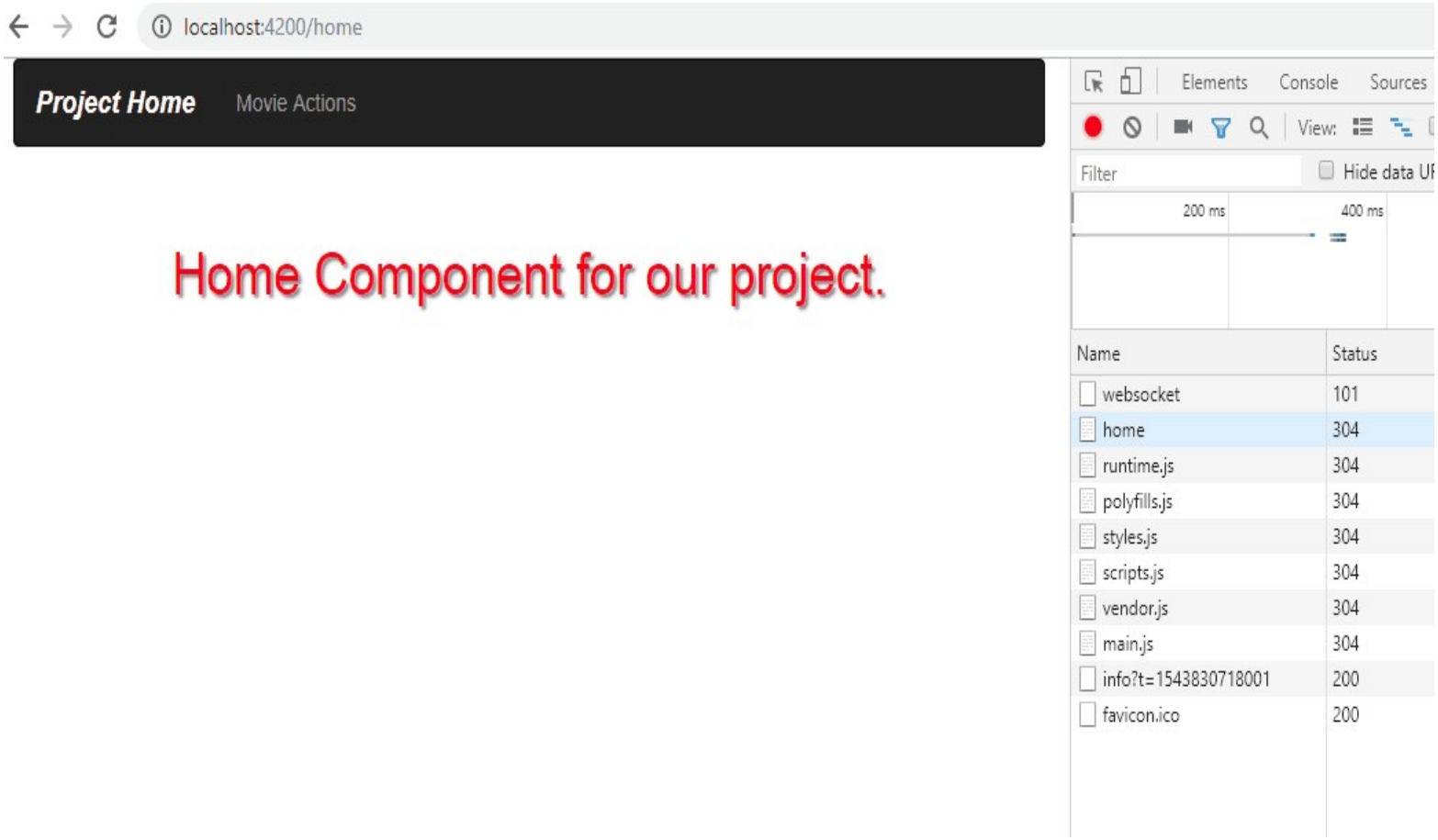
menu.component.html × movie.module.ts movie-routing.module.ts routing.module.ts

```
1   <nav class="navbar navbar-inverse">
2     <div class="container-fluid">
3       <div class="navbar-header">
4         <a class="navbar-brand" [routerLink]="['/home']" routerLinkActive="active"
5           [routerLinkActiveOptions]="{exact: true}">Project Home</a>
6       </div>
7       <ul class="nav navbar-nav">
8         <li><a [routerLink]="['/movie/list']" routerLinkActive="active"
9           [routerLinkActiveOptions]="{exact: true}">Movie Actions</a></li>
10      </ul>
11    </div>
12  </nav>
```

Project Home **Movie Actions**

movies works!

Lazy Loading i Prikaz Podataka



Lazy Loading i Prikaz Podataka

Sada ćemo da dodamo novi interfejs koji će prestavljati Movie model na Angular strani

movie.model.ts

```
export interface Movie{
    id: string,
    name: string,
    genre: string,
    director: string
}
```

U TypeScript-u, interjesi nam omogućavaju da imamo tipizirane podatke svojih custom modela i isto tako daju nam mogućnost intelisensa kada pravimo objekat istog tipa.

Isto tako ukoliko napravimo objekat tipa movie ali ne ubacimo sve zahtevane propertie, dobićemo crvenu liniju kao znak greške, jer nisu obezbeđena sva zahtevana polja.

Lazy Loading i Prikaz Podataka

Da bi smo učitali podatke sa servera, prvo moramo da izmenimo movies.component fajl

movies.component.ts × http.service.ts

```
1   import { Component, OnInit } from '@angular/core';
2   import { Movie } from 'src/app/_interfaces/movie.model';
3   import { HttpService } from 'src/app/shared/http.service';
4
5   @Component({
6     selector: 'app-movies',
7     templateUrl: './movies.component.html',
8     styleUrls: ['./movies.component.css']
9   })
10  export class MoviesComponent implements OnInit {
11    public movies: Movie[] = [];
12
13    constructor(private httpService: HttpService) { }
14
15    ngOnInit() {
16      this.getMovies();
17    }
18
19    public getMovies = () => {
20      let route: string = '/api/movie';
21      this.httpService.getData(route)
22      .subscribe((result) => {
23        this.movies = result as Movie[];
24      },
25      (error) => {
26        console.error(error);
27      });
28    }
29  }
```

Lazy Loading i Prikaz Podataka

Sada ćemo izmeniti i movies.component.html fajl

movies.component.html

```
<div class="row">
  <div class="col-md-offset-10 col-md-2">
    <a href="#">Create owner</a>
  </div>
</div>
<br>
<div class="row">
  <div class="col-md-12">
    <div class="table'responsive">
      <table class="table table-striped">
        <thead>
          <tr>
            <th>Name</th>
            <th>Genre</th>
            <th>Director</th>
            <th>Update</th>
            <th>Delete</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor='let movie of movies'>
            <td>{{movie?.name}}</td>
            <td>{{movie?.genre}}</td>
            <td>{{movie?.director}}</td>
            <td><button type="button" id="update" class="btn btn-success">Update</button></td>
            <td><button type="button" id="delete" class="btn btn-danger">Delete</button></td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>
```

Lazy Loading i Prikaz Podataka

Rezultat

Project Home *Movie Actions*

Create owner

Name	Genre	Director	Update	Delete
Alien: Covenant	Sci-Fi, Thriller	Ridley Scott	Update	Delete
The Lord Of The Rings	Adventure, Drama, Fantasy	Peter Jackson	Update	Delete
World War Z	Action, Adventure, Horror	Marc Forster	Update	Delete
Original Sin	Drama, Mystery, Romance	Michael Cristofer	Update	Delete

Refaktorisanje Komponenti

Uvek trebamo težiti manjim i čitljivijim komponentama, pa ćemo iz tog razloga ovu movies komponentu da razbijemo na dve manje komponente

```
E:\GitHub_Repo\predavanje-netcore-angular\MoviesClient>ng g c movie/movies/movie-table --spec false
CREATE src/app/movie/movies/movie-table/movie-table.component.html (30 bytes)
CREATE src/app/movie/movies/movie-table/movie-table.component.ts (288 bytes)
CREATE src/app/movie/movies/movie-table/movie-table.component.css (0 bytes)
UPDATE src/app/movie/movie.module.ts (482 bytes)
```

Refaktorisanje Komponenti

Movie-table.component.html fajl

movie-table.component.html × | movie-table.component.ts | movies.component.html | movie.module.ts

```
1   <div class="table'responsive">
2     <table class="table table-striped">
3       <thead>
4         <tr>
5           <th>Name</th>
6           <th>Genre</th>
7           <th>Director</th>
8           <th>Update</th>
9           <th>Delete</th>
10        </tr>
11      </thead>
12      <tbody>
13        <tr *ngFor='let movie of movies'>
14          <td>{{movie?.name}}</td>
15          <td>{{movie?.genre}}</td>
16          <td>{{movie?.director}}</td>
17          <td><button type="button" id="update" class="btn btn-success">Update</button></td>
18          <td><button type="button" id="delete" class="btn btn-danger">Delete</button></td>
19        </tr>
20      </tbody>
21    </table>
22  </div>
```

Refaktorisanje Komponenti

Movie-table.component.ts fajl

movie-table.component.ts × movie-table.component.html mo

```
1  import { Component, OnInit, Input } from '@angular/core
2
3  @Component({
4    selector: 'app-movie-table',
5    templateUrl: './movie-table.component.html',
6    styleUrls: ['./movie-table.component.css']
7  })
8  export class MovieTableComponent implements OnInit {
9    @Input() public movies;
10
11   constructor() { }
12
13   ngOnInit() {
14   }
15
16 }
```

@Input dekorator očekuje da mu bude prosledjeno nešto iz roditelj komponente

Refaktorisanje Komponenti

Movies.component.html fajl

```
<div class="row">
  <div class="col-md-offset-10 col-md-2">
    <a href="#">Create owner</a>
  </div>
</div>
<br>
<div class="row">
  <div class="col-md-12">
    <app-movie-table [movies]='movies'></app-movie-table>
  </div>
</div>
```

Posle ovih izmena, rezultat će i dalje biti isti, ali ovaj put uz mnogo lepši i pregledniji kod.

Projektovanje eventa iz child ka parent komponenti

Da bi smo prosledili event iz child komponente ka parent komponenti potreban nam je još jedan dekorator @Output i klasa EventEmitter za prosleđivanje događaja

@Output dekorator služi da bi povezao dve komponente sa eventima

EventEmitter klasa služi da bi se prosledio event preko @Output dekoratora

Izmene ćemo praviti na child komponenti prvo pa zatim na parent komponenti.

Projektovanje eventa iz child ka parent komponenti

Movie-table.component.html fajl

```
<div class="table'responsive">
  <table class="table table-striped">
    <thead>
      <tr>
        <th>Name</th>
        <th>Genre</th>
        <th>Director</th>
        <th>Update</th>
        <th>Delete</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor='let movie of movies'>
        <td>{{movie?.name}}</td>
        <td>{{movie?.genre}}</td>
        <td>{{movie?.director}}</td>
        <td><button type="button" id="update" class="btn btn-success" (click)="update(movie.id)">Update</button></td>
        <td><button type="button" id="delete" class="btn btn-danger" (click)="delete(movie.id)">Delete</button></td>
      </tr>
    </tbody>
  </table>
</div>
```

Projektovanje eventa iz child ka parent komponenti

Movie-table.component.ts fajl

movie-table.component.ts × movie-table.component.html

```
1   import { Component, OnInit, Input, Output, EventEmitte
2
3   @Component({
4     selector: 'app-movie-table',
5     templateUrl: './movie-table.component.html',
6     styleUrls: ['./movie-table.component.css']
7   })
8   export class MovieTableComponent implements OnInit {
9     @Input() public movies;
10    @Output() public onUpdate = new EventEmitter();
11    @Output() public onDelete = new EventEmitter();
12
13    constructor() { }
14
15    ngOnInit() {
16    }
17
18    public update = (id: string) => {
19      this.onUpdate.emit(id);
20    }
21
22    public delete = (id: string) => {
23      this.onDelete.emit(id);
24    }
25  }
```

Projektovanje eventa iz child ka parent komponenti

Movies.component.html fajl

movies.component.html | movie-table.component.ts | movie-table.component.html

```
1   <div class="row">
2     <div class="col-md-offset-10 col-md-2">
3       <a href="#">Create owner</a>
4     </div>
5   </div>
6   <br>
7   <div class="row">
8     <div class="col-md-12">
9       <app-movie-table [movies]='movies' (onUpdate)="redirectToUpdate($event)"
10                       (onDelete)="redirectToDelete($event)"></app-movie-table>
11    </div>
12  </div>
13
```

Projektovanje eventa iz child ka parent komponenti

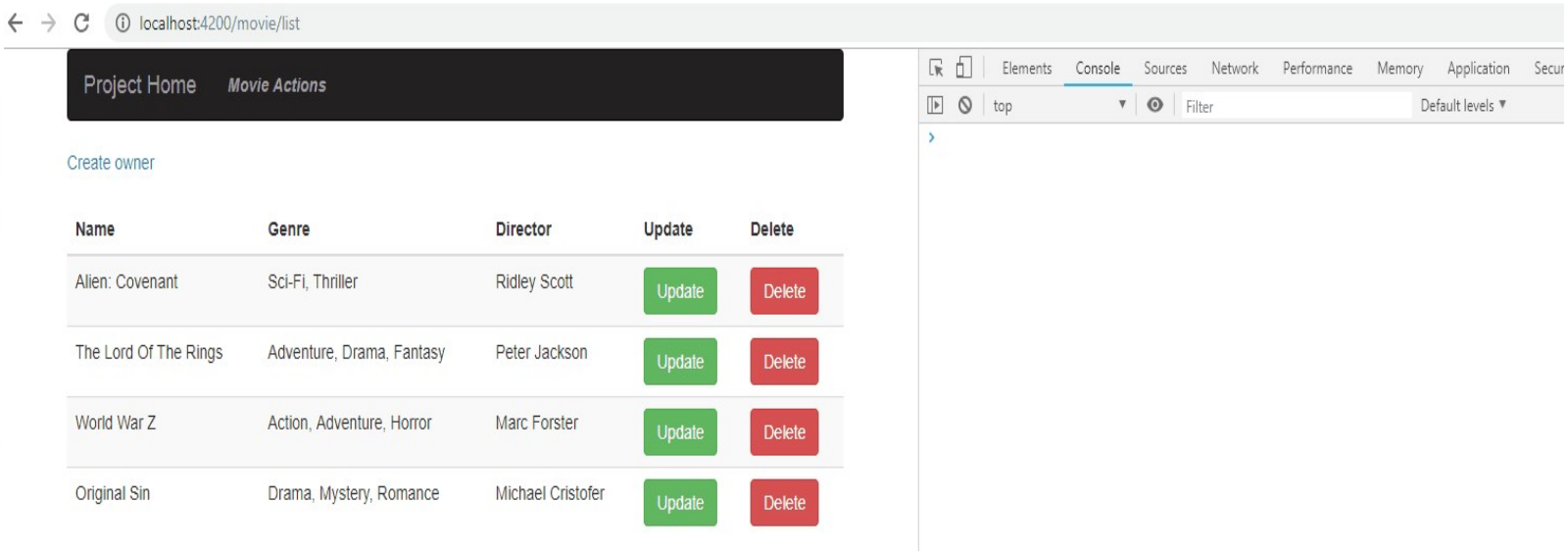
Movies.component.ts fajl

movies.component.ts | movies.component.html | movie-table.component.ts

```
})
export class MoviesComponent implements OnInit {
  public movies: Movie[] = [];

  constructor(private httpService: HttpService) { }

  ngOnInit() {···
  }

  public getMovies = () => {···
  }

  public redirectToUpdate = (event) => {
    console.log('Redirecting to update page with id: ', event)
    //use Router to navigate to required component
  }

  public redirectToDelete = (event) => {
    console.log('Redirecting to delete page with id: ', event)
    //use Router to navigate to required component
  }
}
```

Projektovanje eventa iz child ka parent komponenti

Rezultat



Atribut Direktive u Angularu

Kada želimo da menjamo ponašanje naših html elemenata, najbolji način za to je da koristimo Atribut Direktive

Naglašavam Atribut Direktive zato što imamo još dve vrste direktiva u angularu:

Component Directives – Direktive koje imaju template u sebi

Structural Directives – Direktive za dodavanje i uklanjanje elemenata iz DOM-a : *ngIf i *ngFor

```
<div class="row" *ngIf='owner?.accounts.length <= 2; else advancedUser'>
  <div class="col-md-3">
    <strong>Type of user:</strong>
  </div>
  <div class="col-md-3">
    <span class="text-success">Beginner user.</span>
  </div>
</div>
<ng-template #advancedUser>
  <div class="row">
    <div class="col-md-3">
      <strong>Type of user:</strong>
    </div>
    <div class="col-md-3">
      <span class="text-info">Advanced user.</span>
    </div>
  </div>
</ng-template>
```

Atribut Direktive u Angularu

Pretpostavimo da u našoj tabeli kada pređemo preko svakog header naslova, on promeni boju i oblik pointera i na klik emituje određeni događaj.

```
E:\GitHub_Repo\predavanje-netcore-angular\MoviesClient>ng g directive shared\color-and-emit --spec false --module movie
CREATE src/app/shared/color-and-emit.directive.ts (153 bytes)
UPDATE src/app/movie/movie.module.ts (574 bytes)
```

Kod iz direktive izgleda ovako

Atribut Direktive u Angularu

color-and-emit.directive.ts × | movies.component.ts | movies.component.html

```
1   import { Directive, ElementRef, HostListener } from '@angular/core';
2
3   @Directive({
4     selector: '[appColorAndEmit]'
5   })
6   export class ColorAndEmitDirective {
7
8     constructor(private element: ElementRef) { }
9
10    @HostListener('mouseenter') onMouseEnter() {
11      this.element.nativeElement.style.color = 'blue';
12      this.element.nativeElement.style.cursor = 'pointer'
13    }
14
15    @HostListener('mouseleave') onMouseLeave() {
16      this.element.nativeElement.style.color = 'black';
17      this.element.nativeElement.style.cursor = 'auto'
18    }
19
20    @HostListener('click') onMouseClik(){
21      let text = this.element.nativeElement.innerHTML;
22      console.log("Link ", text , " has been clicked.")
23    }
24
25  }
26
```

Atribut Direktive u Angularu

movie-table.component.html × color-and-emit.directive.ts

```
1   <div class="table'responsive">
2     <table class="table table-striped">
3       <thead>
4         <tr>
5           <th appColorAndEmit>Name</th>
6           <th appColorAndEmit>Genre</th>
7           <th appColorAndEmit>Director</th>
8           <th appColorAndEmit>Update</th>
9           <th appColorAndEmit>Delete</th>
10        </tr>
11      </thead>
```

localhost:4200/movie/list

Project Home *Movie Actions*

Create owner

Name	Genre	Director	Update	Delete
Alien: Covenant	Sci-Fi, Thriller	Ridley Scott	Update	Delete
The Lord Of The Rings	Adventure, Drama, Fantasy	Peter Jackson	Update	Delete

Elements Console Sources Network Performance Memory Application Sec

top Filter Default levels

Angular is running in the development mode. Call enableProdMode() to enable the productio

POST Zahtev i Validacija

Postoje dve vrste validacije: Template i Reactive Validacije

Template validacija se zasniva iskljucivo na template (.html) fajlu.

Kada nije potrebna složena validacija, Template validacija je dobar izbor

Html kod može postati prilično pretrpan i nečitljiv

Reactive validacija se zasniva na kombinacija .html i .ts fajlova i pretežno je usmerena ka .ts fajlu.

Bolja organizacija koda jer se sve dinamički podešava u .ts fajlu

Bolji i čitljiviji kod samog procesa validacije

POST Zahtev i Validacija

Počećemo sa kreiranjem komponente za POST zahtev

```
E:\GitHub_Repo\predavanje-netcore-angular\MoviesClient>ng g c movie/create-movie --spec false
CREATE src/app/movie/create-movie/create-movie.component.html (31 bytes)
CREATE src/app/movie/create-movie/create-movie.component.ts (292 bytes)
CREATE src/app/movie/create-movie/create-movie.component.css (0 bytes)
UPDATE src/app/movie/movie.module.ts (676 bytes)
```

Izmenićemo Movie routing module fajl, da bi smo napravili rutu ka ovoj komponenti

movie.module.ts | movie-routing.module.ts × | app.module.ts

```
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { MoviesComponent } from '../movies/movies.component';
4  import { Routes, RouterModule } from '@angular/router';
5  import { CreateMovieComponent } from '../create-movie/create-movie.component';
6
7  const routes: Routes = [
8    { path: 'list', component: MoviesComponent },
9    { path: 'create', component: CreateMovieComponent}
10 ];
```

POST Zahtev i Validacija

Omogućili smo rutu ka create komponenti i sada treba da obezbedimo i samu navigaciju

Već smo videli u (menu komponenti) kako izgleda attribute routing pomoću [routerLink], pa ćemo sada to uraditi na drugačiji način, koristeći dinamičko rutiranje u kodu.

Izmenićemo prvo movies.component.html fajl

movies.component.html | movie.module.ts | movie-routing.module.ts | movies.comp

```
1  <div class="row">
2    <div class="col-md-offset-10 col-md-2">
3      <a (click)="navigateToCreate()" style="cursor:pointer;">Create owner</a>
4    </div>
5  </div>
```

POST Zahtev i Validacija

A zatim i .ts fajl

movies.component.ts ● | movies.component.html | movie.module.ts | movie-routing.

```
1   import { Component, OnInit } from '@angular/core';
2   import { Movie } from 'src/app/_interfaces/movie.model';
3   import { HttpService } from 'src/app/shared/http.service';
4   import { Router } from '@angular/router';
5
6   @Component({ ···
10  })
11  export class MoviesComponent implements OnInit {
12    public movies: Movie[] = [];
13
14    constructor(private httpService: HttpService, private router: Router) { }
15
16    ngOnInit() { ···
18    }
19
20    public navigateToCreate = () => {
21      this.router.navigate(['/movie/create']);
22    }
```

POST Zahtev i Validacija

Da bi smo koristili Reactive validaciju, moramo indžektovati ReactiveFormsModule

movie.module.ts × | movies.component.ts | movies.component.html | movie-routing.module.ts

```
1   import { NgModule } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3   import { ReactiveFormsModule } from '@angular/forms';
4
5   import { MoviesComponent } from './movies/movies.component';
6   import { MovieRoutingModule } from './movie-routing/movie-routing.module';
7   import { MovieTableComponent } from './movies/movie-table/movie-table.component';
8   import { ColorAndEmitDirective } from '../shared/color-and-emit.directive';
9   import { CreateMovieComponent } from './create-movie/create-movie.component';
10
11  @NgModule({
12    declarations: [
13    MoviesComponent,
14    MovieTableComponent,
15    ColorAndEmitDirective,
16    CreateMovieComponent
17  ],
18    imports: [
19      CommonModule,
20      MovieRoutingModule,
21      ReactiveFormsModule
22    ]
23  })
24  export class MovieModule { }
```

POST Zahtev i Validacija

Kreiraćemo i jedan interfejs za potrebe kreiranja našeg objekta

movieForCreation.model.ts | movie.module.ts

```ts
export interface MovieForCreation {
    name: string,
    genre: string,
    director: string
}
```

I sada ćemo preći na kreiranje logike za POST zahtev i samu validaciju

create-movie.component.html | create-movie.component.ts

```html
<div class="container-fluid">
  <form [formGroup]="movieForm" autocomplete="off" novalidate (ngSubmit)="createMovie(movieForm.value)">

  </form>
</div>
```

POST Zahtev i Validacija

Trebamo obezbediti formGroup u .ts fajlu

create-movie.component.ts × | movie-routing.module.ts | movies.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms';
import { Location } from '@angular/common';
import { MovieForCreation } from 'src/app/_interfaces/movieForCreation.model';
import { HttpService } from 'src/app/shared/http.service';

@Component({
  selector: 'app-create-movie',
  templateUrl: './create-movie.component.html',
  styleUrls: ['./create-movie.component.css']
})
export class CreateMovieComponent implements OnInit {
  public movieForm: FormGroup;

  constructor(private location: Location, private httpService: HttpService) { }

  ngOnInit() {
    this.movieForm = new FormGroup({
      name: new FormControl('', [Validators.required, Validators.maxLength(40)]),
      genre: new FormControl('', [Validators.required, Validators.maxLength(20)]),
      director: new FormControl('', [Validators.required, Validators.maxLength(20)])
    });
  }
```

POST Zahtev i Validacija

Sada možemo nastaviti sa .html fajlom

create-movie.component.html × create-movie.component.ts

```
1  <div class="container-fluid">
2    <form [formGroup]="movieForm" autocomplete="off" novalidate (ngSubmit)="createMovie(movieForm.value)">
3      <div class="form-horizontal well">
4
5        <div class="form-group">
6          <label for="name" class="control-label col-md-2">Name of a movie: </label>
7          <div class="col-md-5">
8            <input type="text" formControlName="name" id="name" class="form-control" />
9          </div>
10         <div class="col-md-5">
11           <em *ngIf="isInvalid('name') && hasError('name', 'required')">Name is required</em>
12           <em *ngIf="isInvalid('name') && hasError('name', 'maxlength')">Maximum allowed length is 40 characters.</em>
13         </div>
14       </div>
15
16     </div>
17   </form>
18 </div>
```

POST Zahtev i Validacija

Napravićemo ove dve funkcije u .ts fajlu

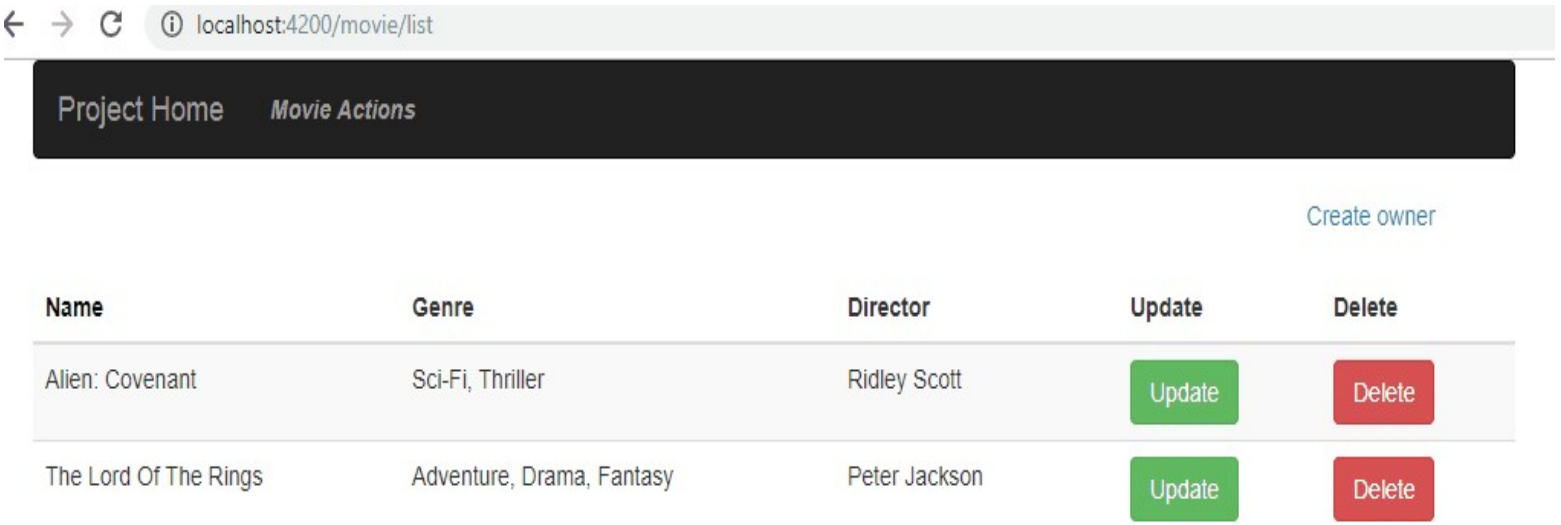
```ts
public isInvalid = (controlName: string) => {
  let control = this.movieForm.controls[controlName];
  return control.invalid && control.touched;
}

public hasError = (controlName: string, errorName: string) => {
  return this.movieForm.controls[controlName].hasError(errorName);
}
```

styles.css × create-movie.compone

```css
em{
    color: #e71515;
    font-weight: bold;
}

.ng-invalid.ng-touched{
    border-color: red;
}
```

POST Zahtev i Validacija

Trenutni rezultat



POST Zahtev i Validacija

Ispod validacije za name napravićemo još dve kontrole sa validacijom

```
<div class="form-group">
  <label for="genre" class="control-label col-md-2">Genre of a movie: </label>
  <div class="col-md-5">
    <input type="text" formControlName="genre" id="genre" class="form-control" />
  </div>
  <div class="col-md-5">
    <em *ngIf="isInvalid('genre') && hasError('genre', 'required')">Genre is required</em>
    <em *ngIf="isInvalid('genre') && hasError('genre', 'maxlength')">Maximum allowed length is 20 characters.</em>
  </div>
</div>

<div class="form-group">
  <label for="director" class="control-label col-md-2">Director of a movie: </label>
  <div class="col-md-5">
    <input type="text" formControlName="director" id="director" class="form-control" />
  </div>
  <div class="col-md-5">
    <em *ngIf="isInvalid('director') && hasError('director', 'required')">Director is required</em>
    <em *ngIf="isInvalid('director') && hasError('director', 'maxlength')">Maximum allowed length is 40 characters.</em>
  </div>
</div>
```

POST Zahtev i Validacija

Treba da dodamo i dugmiće

```
<br><br>

<div class="form-group">
  <div class="col-md-offset-5 col-md-1">
    <button type="submit" class="btn btn-info" [disabled]="!movieForm.valid">Save</button>
  </div>
  <div class="col-md-1">
    <button type="button" class="btn btn-danger" (click)="redirectToMovieList()">Cancel</button>
  </div>
</div>
```

Poslednje logika za kreiranje entiteta i za redirekciju na Cancel click

POST Zahtev i Validacija

create-movie.component.ts × create-movie.component.html

```
14
15    constructor(private location: Location, private httpService: HttpService) { }
16
17    ngOnInit() {···
23    }
24
25    public createMovie = (movieFormValue) => {
26      if (this.movieForm.valid) {
27        this.executeMovieCreation(movieFormValue);
28      }
29    }
30
31    private executeMovieCreation = (movieFormValue) => {
32      let movie: MovieForCreation = {
33        name: movieFormValue.name,
34        genre: movieFormValue.genre,
35        director: movieFormValue.director
36      };
37
38      let apiUrl = 'api/movie';
39      this.httpService.create(apiUrl, movie)
40      .subscribe(res => {
41        this.location.back();
42        //modal or something in here to inform user about successfull action
43      },
44      error => {
45        console.log("Error while creating movie ", error);
46      })
47    }
48
49    public redirectToMovieList = () => {
50      this.location.back();
51      //this.router.navigate(['/movie/list']); it could be done this way as well if using Router
52    }
```

POST Zahtev i Validacija

localhost:4200/movie/list

Project Home *Movie Actions*

Create owner

Name	Genre	Director	Update	Delete
Alien: Covenant	Sci-Fi, Thriller	Ridley Scott	Update	Delete
The Lord Of The Rings	Adventure, Drama, Fantasy	Peter Jackson	Update	Delete
World War Z	Action, Adventure, Horror	Marc Forster	Update	Delete
Original Sin	Drama, Mystery, Romance	Michael Cristofer	Update	Delete

Dodatni materijal

Update https://code-maze.com/net-core-web-development-part14/

Delete https://code-maze.com/net-core-web-development-part15/

IIS Deploy https://code-maze.com/net-core-web-development-part16/

Linux Deploy https://code-maze.com/net-core-web-development-part17/

Komplet Serijal https://code-maze.com/angular-series/

Angular Material Serijal https://code-maze.com/angular-material-series/

Angual Best Practices https://code-maze.com/angular-best-practices/

JWT .NET Core and Angular (obuhvaćen i Angular Route Gurad) https://code-maze.com/authentication-aspnetcore-jwt-1/ https://code-maze.com/authentication-aspnetcore-jwt-2/