Развој на серверски WEB апликации

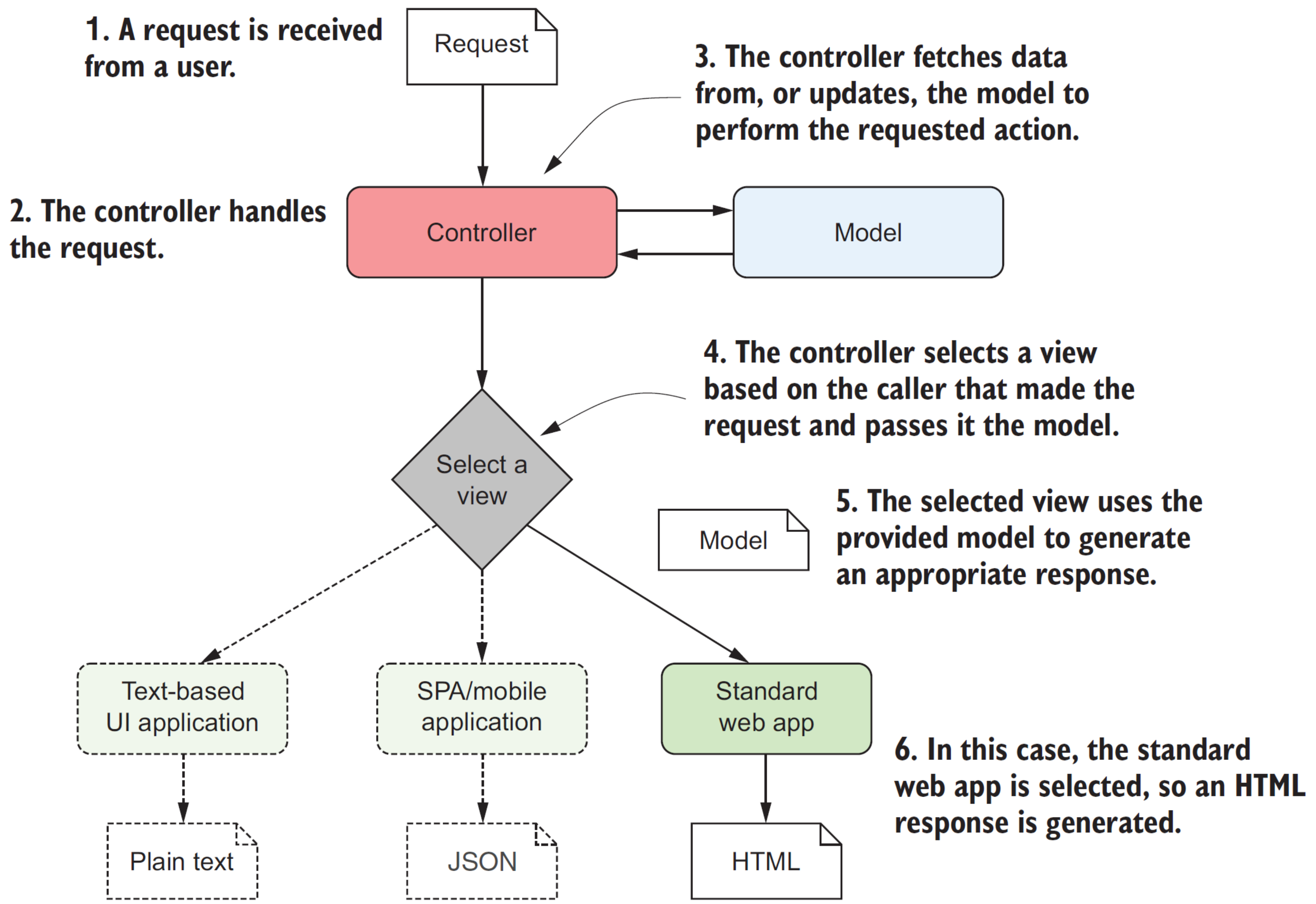
ASP.NET Core MVC

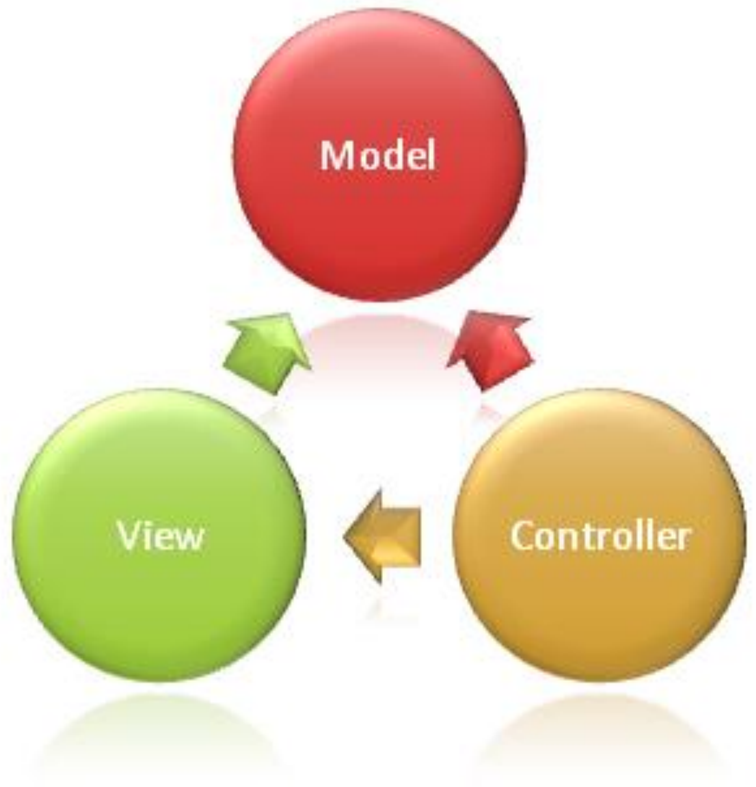
The MVC design pattern (1)

- The Model-View-Controller (MVC) architectural pattern separates an application into three main groups of components:
  - **Model**—The data that needs to be displayed, the state of the application (i.e., business logic of the application).
  - **View**—The template that displays the data provided by the model (i.e., user interface presenting the content).
  - **Controller**—Updates the model and selects the appropriate view (i.e., input logic of the application).

- In general, the order of events when an application responds to a user interaction or request is as follows:
  1. The controller receives the request;
  2. Depending on the request, the controller either fetches the requested data from the application model, or it updates the data that makes up the model;
  3. The controller selects a view to display and passes the model to it;
  4. The view uses the data contained in the model to generate the UI.

The MVC design pattern (2)

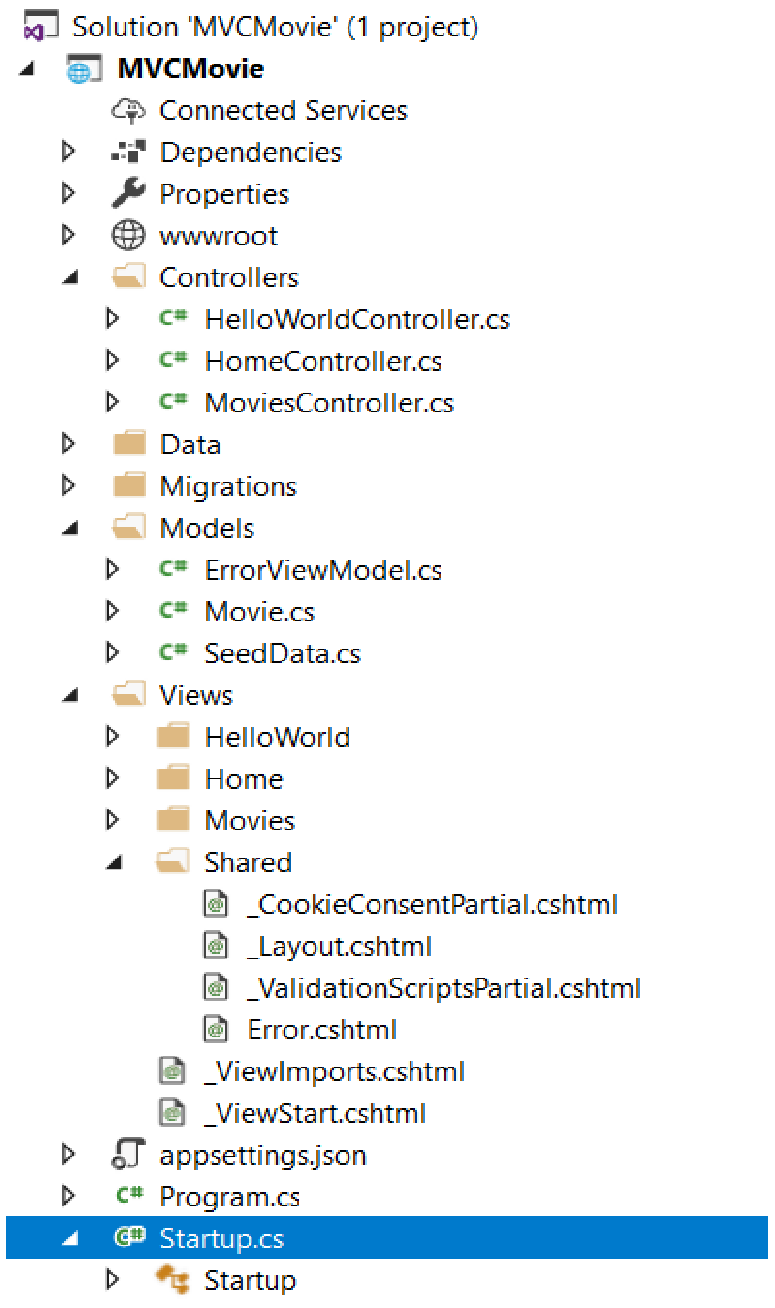

1. A request is received from a user.
Request
3. The controller fetches data from, or updates, the model to perform the requested action.
2. The controller handles the request.
Controller
Model
4. The controller selects a view based on the caller that made the request and passes it the model.
Select a view
Model
5. The selected view uses the provided model to generate an appropriate response.
Text-based UI application
SPA/mobile application
Standard web app
6. In this case, the standard web app is selected, so an HTML response is generated.
Plain text
JSON
HTML



MVC project structure

- Models folder and subfolders
  - contain all the models in the project
  - C# classes
- Views folder and subfolders
  - contain all the views in the project
  - cshtml Razor Views
- Controllers folder and subfolder
  - contain all the controllers in the project
  - C# classes inheriting from the Controller base class

- The application is configured to use MVC in Configure method (Startup.cs)



```
Solution 'MVCMovie' (1 project)
  MVCMovie
    Connected Services
    Dependencies
    Properties
    wwwroot
    Controllers
      HelloWorldController.cs
      HomeController.cs
      MoviesController.cs
    Data
    Migrations
    Models
      ErrorViewModel.cs
      Movie.cs
      SeedData.cs
    Views
      HelloWorld
      Home
      Movies
      Shared
        _CookieConsentPartial.cshtml
        _Layout.cshtml
        _ValidationScriptsPartial.cshtml
        Error.cshtml
      _ViewImports.cshtml
      _ViewStart.cshtml
    appsettings.json
    Program.cs
    Startup.cs
      Startup
```

Models

- An MVC model is a collection of classes.
- When you create a model class, you define the properties and methods that are required for the kind of object the model class describes.
- It is also important to know how controllers pass models to views, and how views can render the data stored in a model to the browser.
- Models often include data access logic that reads data from a database and writes data to that database.
- Models are defined in the Models folder of the project.

Example model Movie.cs

```
using System;
using System.ComponentModel.DataAnnotations;

namespace MVCMovie.Models
{
    public class Movie
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public DateTime ReleaseDate { get; set; }
        public string Genre { get; set; }
        public decimal Price { get; set; }
    }
}
```

Model Validators (DataAnnotations)

- Here are some of the built-in validation attributes:
  - [**CreditCard**]: Validator for credit card format.
  - [**Compare**]: Validates that two properties in a model match.
  - [**EmailAddress**]: Validator for email format.
  - [**Phone**]: Validator for telephone number format.
  - [**Range**]: Validates that the property value falls within a specified range.
  - [**RegularExpression**]: Validates that the property value matches a specified regular expression.
  - [**Required**]: Validates that the field is not null.
  - [**StringLength**]: Validates that a string property value doesn't exceed a specified length limit.
  - [**Url**]: Validates that the property has a URL format.
  - [**DateTime**]: Validator for datetime format.
- A complete list of validation attributes can be found in the System.ComponentModel.DataAnnotations namespace.

Model Validators (2)

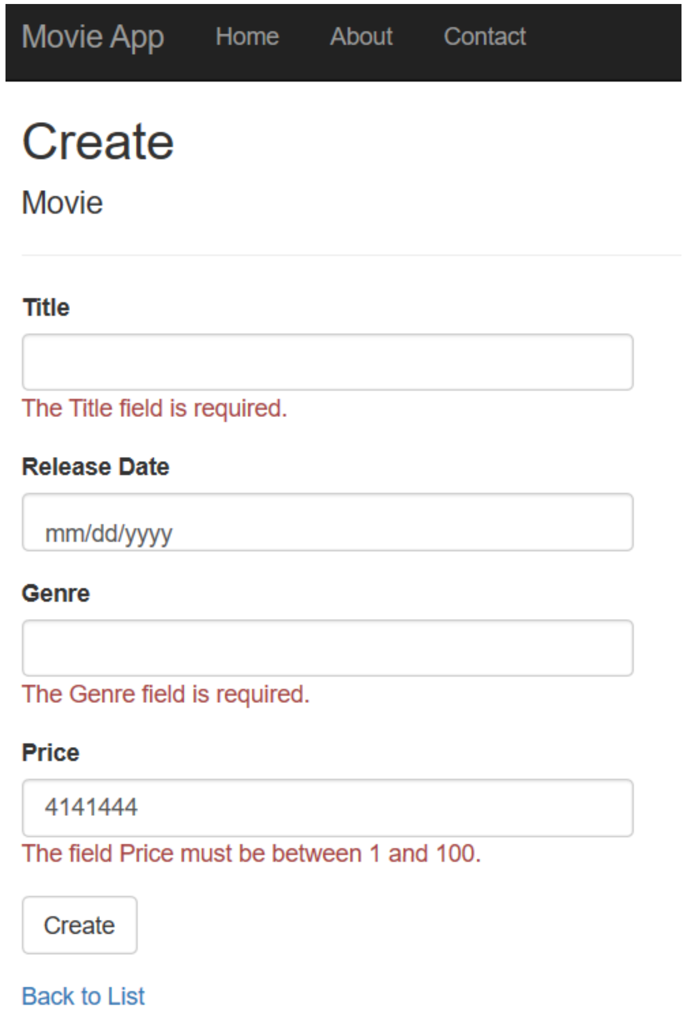
```
namespace MVCMovie.Models
{
    public class Movie
    {
        public int Id { get; set; }

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

        [Display(Name = "Release Date")]
        [DataType(DataType.Date)]
        public DateTime? ReleaseDate { get; set; }

        [RegularExpression(@"^[A-Z]+[a-zA-Z""'\s-]*$")]
        [Required]
        [StringLength(30)]
        public string Genre { get; set; }

        [Range(1, 100)]
        [DataType(DataType.Currency)]
        public decimal? Price { get; set; }
    }
}
```



Create

Movie

Title

The Title field is required.

Release Date

mm/dd/yyyy

Genre

The Genre field is required.

Price

4141444

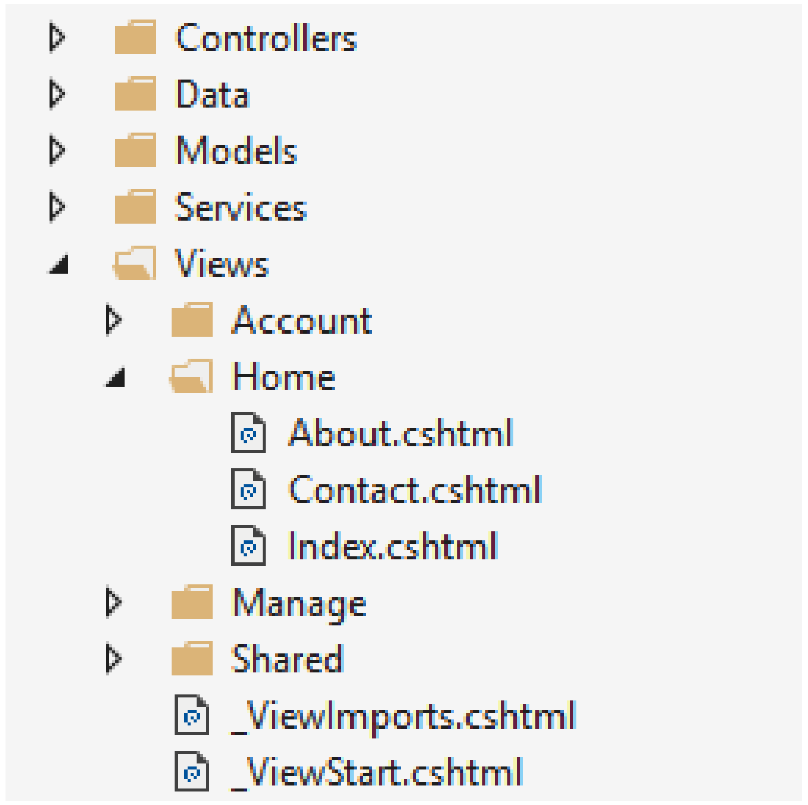
The field Price must be between 1 and 100.

Views

- In the Model-View-Controller (MVC) pattern, the view handles the app's data presentation and user interaction.
- A view is an HTML template with embedded Razor markup.
  - Razor markup is code that interacts with HTML markup to produce a **rendered HTML webpage** that's sent to the client.
- Razor views in MVC can be strongly typed based on your model.
- Controllers can pass a strongly typed model to views enabling your views to have type checking and IntelliSense support.

Views

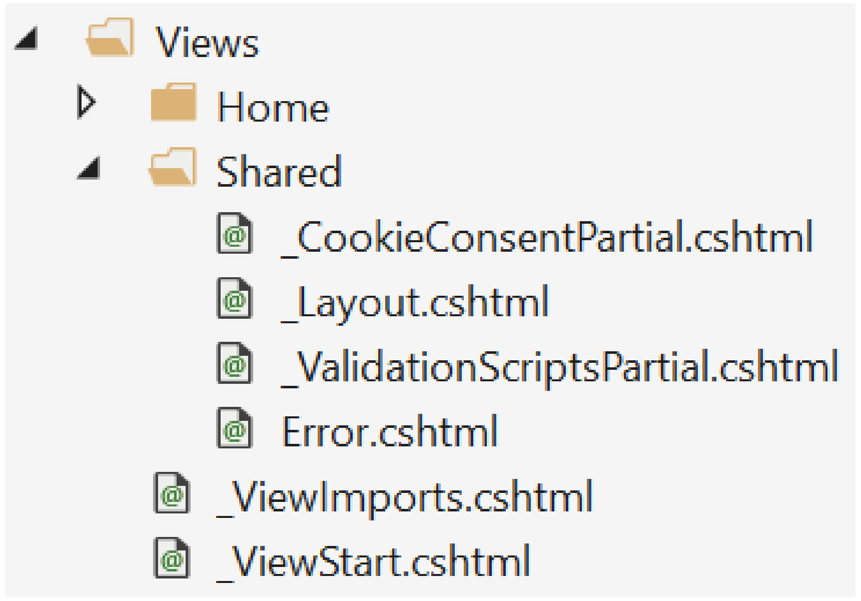
- Usually, view files are grouped into folders named for each of the app's controllers. The folders are stored in a Views folder at the root of the app.
- A Home Controller is represented by/corresponds to a Home folder inside the Views folder.
- The Home folder contains the views for the **About**, **Contact** and **Index** webpages.
- When a user requests one of these three webpages, controller actions in the Home Controller determine which of the three views is used to build and return a webpage to the user.



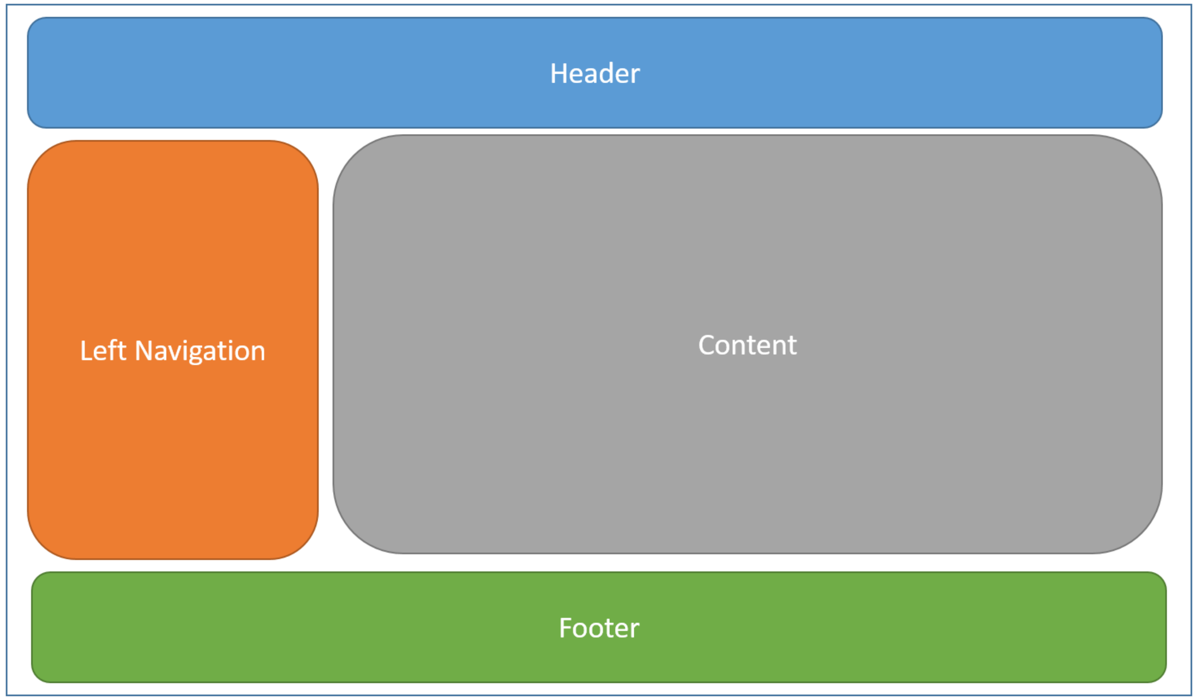
```
▷ Controllers
▷ Data
▷ Models
▷ Services
◢ Views
    ▷ Account
    ◢ Home
        About.cshtml
        Contact.cshtml
        Index.cshtml
    ▷ Manage
    ▷ Shared
      _ViewImports.cshtml
      _ViewStart.cshtml
```

Using layouts

- Use layouts to provide consistent webpage sections and reduce code repetition.
- Layouts often contain the header, navigation and menu elements, and the footer.



```
Views
    Home
    Shared
        _CookieConsentPartial.cshtml
        _Layout.cshtml
        _ValidationScriptsPartial.cshtml
        Error.cshtml
    _ViewImports.cshtml
    _ViewStart.cshtml
```



_Layout.cshtml (1)

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Movie App</title>

    <environment include="Development">
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
        <link rel="stylesheet" href="~/css/site.css" />
    </environment>
    <environment exclude="Development">
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css"
              asp-fallback-href="~/lib/bootstrap/dist/css/bootstrap.min.css"
              asp-fallback-test-class="sr-only" asp-fallback-test-property="position" asp-fallback-test-value="absolute" />
        <link rel="stylesheet" href="~/css/site.min.css" asp-append-version="true" />
    </environment>
</head>
<body>
    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a asp-area="" asp-controller="Movies" asp-action="Index" class="navbar-brand">Movie App</a>
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li><a asp-area="" asp-controller="Home" asp-action="Index">Home</a></li>
                    <li><a asp-area="" asp-controller="Home" asp-action="About">About</a></li>
                    <li><a asp-area="" asp-controller="Home" asp-action="Contact">Contact</a></li>
                </ul>
            </div>
        </div>
    </nav>
```

_Layout.cshtml (2)

```
<partial name="_CookieConsentPartial" />

    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; 2020 - Movie App - <a asp-controller="Home" asp-action="Privacy">Privacy</a></p>
        </footer>
    </div>

    <environment include="Development">
        <script src="~/lib/jquery/dist/jquery.js"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.js"></script>
        <script src="~/js/site.js" asp-append-version="true"></script>
    </environment>
    <environment exclude="Development">
        <script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.3.1.min.js"
                asp-fallback-src="~/lib/jquery/dist/jquery.min.js"
                asp-fallback-test="window.jQuery"
                crossorigin="anonymous"
                integrity="sha384-tsQFqpEReu7ZLhBV2VZlAu7zcOV+rXbYlF2cqB8txI/8aZajjp4Bqd+V6D5IgvKT">
        </script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"
                asp-fallback-src="~/lib/bootstrap/dist/js/bootstrap.min.js"
                asp-fallback-test="window.jQuery && window.jQuery.fn && window.jQuery.fn.modal"
                crossorigin="anonymous"
                integrity="sha384-aJ21OjlMXNL5UyIl/XNwTMqvzeRMZH2w8c5cRVpzpU8Y5bApTppSuUkhZXN0VxHd">
        </script>
        <script src="~/js/site.min.js" asp-append-version="true"></script>
    </environment>

    @RenderSection("Scripts", required: false)
</body>
</html>
```

Index.cshtml

```
@model IEnumerable<MVCMovie.Models.Movie>

@{
    ViewData["Title"] = "Index";
}

<h2>Index</h2>

<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Title)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ReleaseDate)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Genre)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Price)
            </th>
            <th></th>
        </tr>
    </thead>
```

```
 <tbody>
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.ReleaseDate)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Genre)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Price)
            </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>
```

Rendered HTML for the Index view

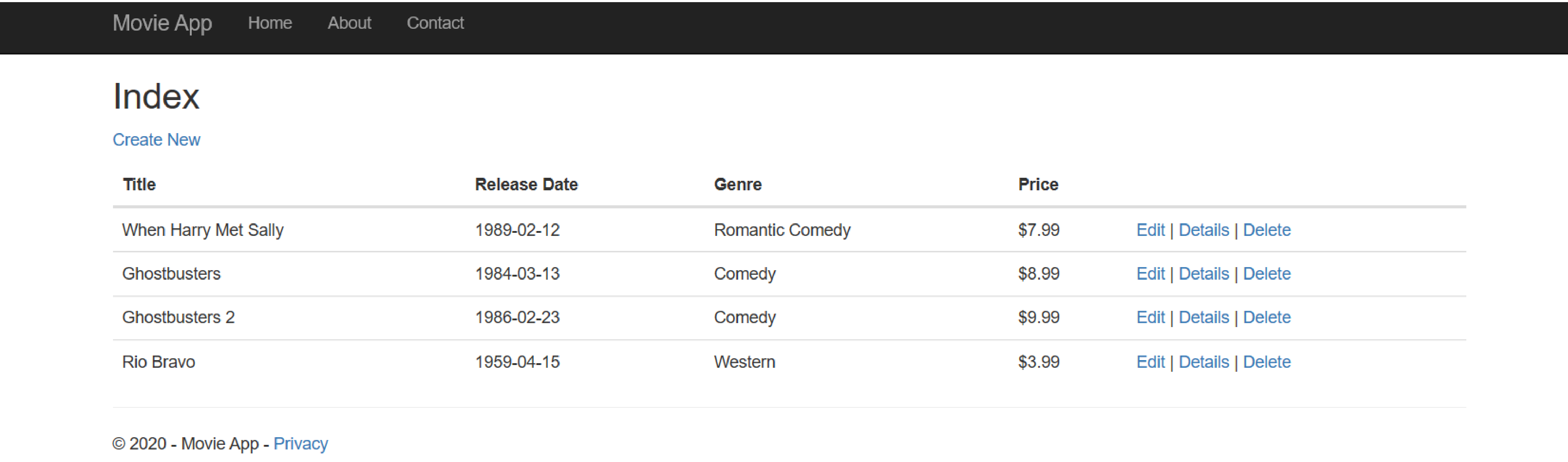
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Index - Movie App</title>
    <link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.css" />
    <link rel="stylesheet" href="/css/site.css" />

</head>
<body>
    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target=".navbar-collapse">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="/Movies">Movie App</a>
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li><a href="/Home/Index">Home</a></li>
                    <li><a href="/Home/About">About</a></li>
                    <li><a href="/Home/Contact">Contact</a></li>
                </ul>
            </div>
        </div>
    </nav>
```

```
   <div class="container body-content">
        <h2>Index</h2>

        <p> <a href="/Movies/Create">Create New</a> </p>
        <table class="table">
            <thead>
                <tr>
                    <th> Title </th>
                    <th> Release Date </th>
                    <th> Genre </th>
                    <th> Price </th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td> When Harry Met Sally </td>
                    <td> 1989-02-12 </td>
                    <td> Romantic Comedy </td>
                    <td> $7.99 </td>
                    <td>
                        <a href="/Movies/Edit/1">Edit</a> |
                        <a href="/Movies/Details/1">Details</a> |
                        <a href="/Movies/Delete/1">Delete</a>
                    </td>
                </tr>
                ...
            </tbody>
        </table>
        ...
   </div>
</body>
</html>
```

Browser view for the Index page



Index

Create New

Title	Release Date	Genre	Price	
When Harry Met Sally	1989-02-12	Romantic Comedy	$7.99	Edit | Details | Delete
Ghostbusters	1984-03-13	Comedy	$8.99	Edit | Details | Delete
Ghostbusters 2	1986-02-23	Comedy	$9.99	Edit | Details | Delete
Rio Bravo	1959-04-15	Western	$3.99	Edit | Details | Delete

© 2020 - Movie App - Privacy

Controllers

- A controller is used to define and group a set of actions.
- An action (or action method) is a method on a controller which handles requests.
- Controllers logically group similar actions together.
- This aggregation of actions allows common sets of rules, such as routing, caching, and authorization, to be applied collectively.
- Requests are mapped to actions through routing.

Example controller

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using MVCMovie.Models;

namespace MVCMovie.Controllers
{
    public class MoviesController : Controller
    {
        private readonly MVCMovieContext _context;

        public MoviesController(MVCMovieContext context)
        { _context = context; }

        // GET: Movies
        public async Task<IActionResult> Index()
        {
            return View(await _context.Movie.ToListAsync());
        }

        // GET: Movies/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null) { return NotFound(); }

            var movie = await _context.Movie
              .FirstOrDefaultAsync(m => m.Id == id);
            if (movie == null) { return NotFound(); }

            return View(movie);
        }
```

```
        // POST: Movies/Create
        [HttpPost]
        public async Task<IActionResult>
Create([Bind("Id,Title,ReleaseDate,Genre,Price")] Movie movie)
        {
            if (ModelState.IsValid) {
                _context.Add(movie);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(movie);
        }

        // POST: Movies/Edit/5
        [HttpPost]
        public async Task<IActionResult> Edit(int id,
[Bind("Id,Title,ReleaseDate,Genre,Price")] Movie movie)
        {
            if (id != movie.Id) { return NotFound(); }

            if (ModelState.IsValid) {
                try {
                    _context.Update(movie);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException) {
                    if (!MovieExists(movie.Id)) { return NotFound(); }
                    else { throw; }
                }
                return RedirectToAction(nameof(Index));
            }
            return View(movie);
        }
        ...
    }
}
```

Views for controller actions (Edit.cshtml and Details.cshtml)

```
@model MVCMovie.Models.Movie

@{
    ViewData["Title"] = "Edit";
}

<h2>Edit</h2>

<h4>Movie</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="Id" />
            <div class="form-group">
                <label asp-for="Title" class="control-label"></label>
                <input asp-for="Title" class="form-control" />
                <span asp-validation-for="Title" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ReleaseDate" class="control-label"></label>
                <input asp-for="ReleaseDate" class="form-control" />
                <span asp-validation-for="ReleaseDate" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Genre" class="control-label"></label>
                <input asp-for="Genre" class="form-control" />
                <span asp-validation-for="Genre" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Price" class="control-label"></label>
                <input asp-for="Price" class="form-control" />
                <span asp-validation-for="Price" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </form>
    </div>
</div>
```

```
@model MVCMovie.Models.Movie

@{
    ViewData["Title"] = "Details";
}

<h2>Details</h2>

<div>
    <h4>Movie</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt> @Html.DisplayNameFor(model => model.Title) </dt>
        <dd> @Html.DisplayFor(model => model.Title) </dd>
        <dt> @Html.DisplayNameFor(model => model.ReleaseDate) </dt>
        <dd> @Html.DisplayFor(model => model.ReleaseDate) </dd>
        <dt> @Html.DisplayNameFor(model => model.Genre) </dt>
        <dd> @Html.DisplayFor(model => model.Genre) </dd>
        <dt> @Html.DisplayNameFor(model => model.Price) </dt>
        <dd> @Html.DisplayFor(model => model.Price) </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.Id">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>
```

Routing to controller actions

- ASP.NET Core MVC uses the Routing middleware to match the URLs of incoming requests and map them to actions.
- Routes describe how URL paths should be matched to actions.
- Actions are either **conventionally routed** (specified in Startup.cs, Configure method, left) or **attribute routed** (specified in the Controller, right)

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

```
public class HomeController : Controller
{
    [Route("")]
    [Route("Home")]
    [Route("Home/Index")]
    public IActionResult Index() { return View(); }
... }
```

- Passing parameters in URL query string mapped to action method parameters (casing and order not important)
  - https://localhost:{PORT}/HelloWorld/Welcome/3?name=Daniel&numtimes=4

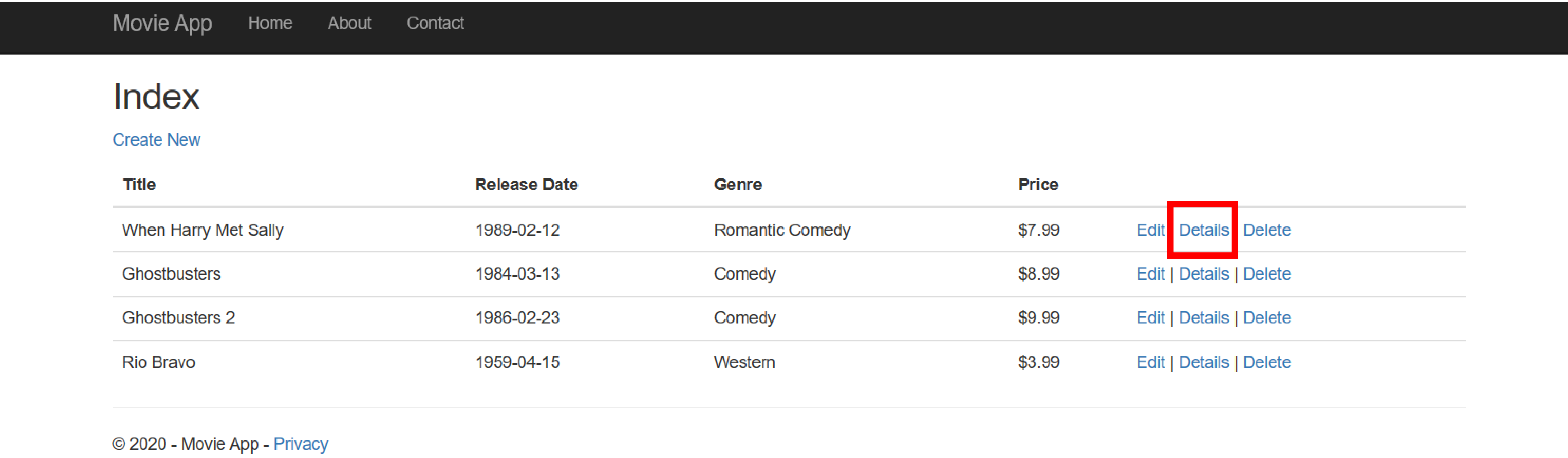
```
public string Welcome(string name, int numTimes = 1, int id = 1)
{ return HtmlEncoder.Default.Encode($"Hello {name}, NumTimes is {numTimes}, id is {id}"); }
```

Controller Helper Methods

- Controllers usually inherit from Controller, although this isn't required. Deriving from Controller provides access to three categories of helper methods:
  - Methods resulting in an **empty response body**
    - *HTTP Status Code:* BadRequest, NotFound and Ok (e.g., `return NotFound();`)
    - *Redirect: Redirect, LocalRedirect, RedirectToAction or RedirectToRoute*
  - Methods resulting in a non-empty response body with a **predefined content type**
    - *View:* `return View(movie);`
    - *Formatted Response:* `return Json(movie);`
  - Methods resulting in a non-empty response body formatted in a **content type negotiated with the client**
    - `return BadRequest(ModelState); return Ok(value);`
      `return CreatedAtRoute("routename", value, newobject);`

MVC workflow example (1)

- User clicks on the Details button on the first movie entry



Movie App Home About Contact

Index

Create New

Title	Release Date	Genre	Price	
When Harry Met Sally	1989-02-12	Romantic Comedy	$7.99	Edit | Details | Delete
Ghostbusters	1984-03-13	Comedy	$8.99	Edit | Details | Delete
Ghostbusters 2	1986-02-23	Comedy	$9.99	Edit | Details | Delete
Rio Bravo	1959-04-15	Western	$3.99	Edit | Details | Delete

© 2020 - Movie App - Privacy

MVC workflow example (2)

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Index - Movie App</title>
    <link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.css" />
    <link rel="stylesheet" href="/css/site.css" />

</head>
<body>
    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target=".navbar-collapse">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="/Movies">Movie App</a>
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li><a href="/Home/Index">Home</a></li>
                    <li><a href="/Home/About">About</a></li>
                    <li><a href="/Home/Contact">Contact</a></li>
                </ul>
            </div>
        </div>
    </nav>
```

```
    <div class="container body-content">
        <h2>Index</h2>
        <p> <a href="/Movies/Create">Create New</a> </p>
        <table class="table">
            <thead>
                <tr>
                    <th> Title </th>
                    <th> Release Date </th>
                    <th> Genre </th>
                    <th> Price </th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td> When Harry Met Sally </td>
                    <td> 1989-02-12 </td>
                    <td> Romantic Comedy </td>
                    <td> $7.99 </td>
                    <td>
                        <a href="/Movies/Edit/1">Edit</a> |
                        <a href="/Movies/Details/1">Details</a> |
                        <a href="/Movies/Delete/1">Delete</a>
                    </td>
                </tr>
                ...
            </tbody>
        </table>
        ...
    </div>
</body>
</html>
```

Rendered HTML of the page

MVC workflow example (3)

- The Details button navigates to `/Movies/Details/1`
- The Application uses conventional routes
  - `routes.MapRoute("default", "{controller=Home}/{action=Index}/{id?}");`

➔ MoviesController is the selected Controller

➔ Details is the selected Action method

➔ id is 1

MVC workflow example (4)

- The MoviesController executes the Details method with id=1

```
namespace MVCMovie.Controllers
{
    public class MoviesController : Controller
    {
        private readonly MVCMovieContext _context;
        ...
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null) { return NotFound(); }

            var movie = await _context.Movie
            .FirstOrDefaultAsync(m => m.Id == id);
            if (movie == null) { return NotFound(); }

            return View(movie);
        }
        ...
    }
}
```

- The action returns the Details.cshml view passing the movie entry

MVC workflow example (5)

- The Details.cshtml is a Razor view that receives the model entry

```
@model MVCMovie.Models.Movie
@{
    ViewData["Title"] = "Details";
}

<h2>Details</h2>
<div>
    <h4>Movie</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt> @Html.DisplayNameFor(model => model.Title) </dt>
        <dd> @Html.DisplayFor(model => model.Title) </dd>
        <dt> @Html.DisplayNameFor(model => model.ReleaseDate) </dt>
        <dd> @Html.DisplayFor(model => model.ReleaseDate) </dd>
        <dt> @Html.DisplayNameFor(model => model.Genre) </dt>
        <dd> @Html.DisplayFor(model => model.Genre) </dd>
        <dt> @Html.DisplayNameFor(model => model.Price) </dt>
        <dd> @Html.DisplayFor(model => model.Price) </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.Id">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>
```

MVC workflow example (6)

- The Razor view is rendered into plain HTML (within the layout)

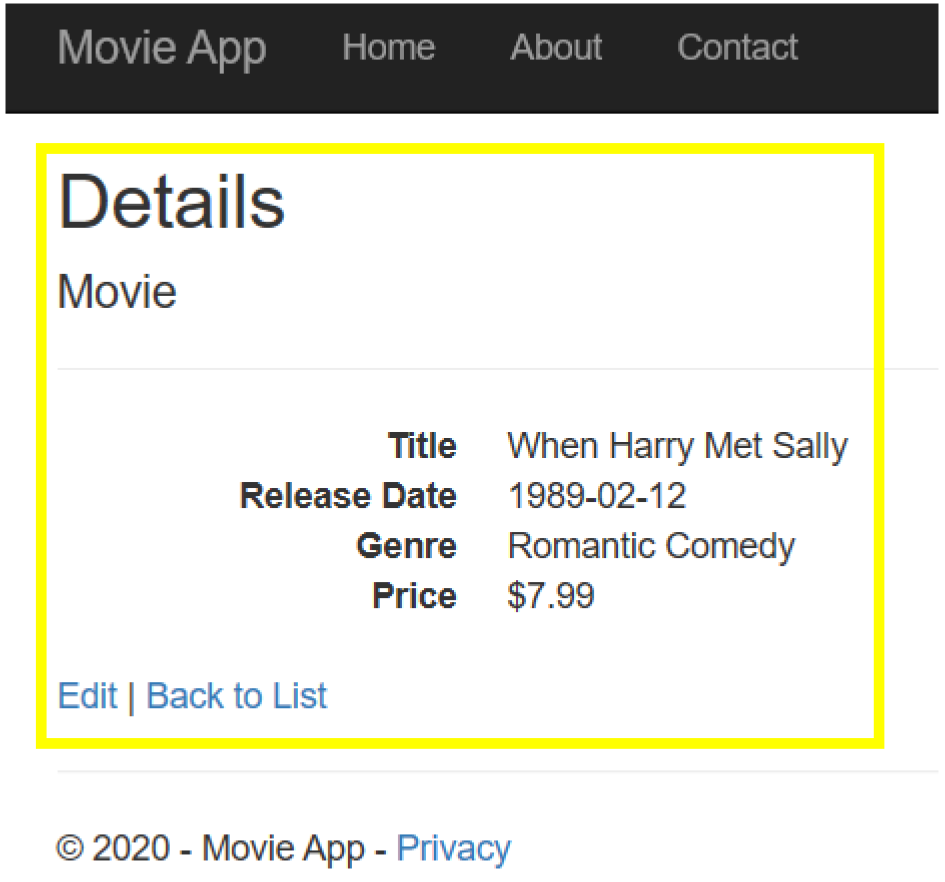
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Details - Movie App</title>
    <link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.css" />
    <link rel="stylesheet" href="/css/site.css" />
</head>
<body>
    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse"
data-target=".navbar-collapse">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="/Movies">Movie App</a>
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li><a href="/Home/Index">Home</a></li>
                    <li><a href="/Home/About">About</a></li>
                    <li><a href="/Home/Contact">Contact</a></li>
                </ul>
            </div>
        </div>
    </nav>
```

```
    <div class="container body-content">
        <h2>Details</h2>
        <div>
            <h4>Movie</h4>
            <hr />
            <dl class="dl-horizontal">
                <dt> Title </dt>
                <dd> When Harry Met Sally </dd>
                <dt> Release Date </dt>
                <dd> 1989-02-12 </dd>
                <dt> Genre </dt>
                <dd> Romantic Comedy </dd>
                <dt> Price </dt>
                <dd> $7.99 </dd>
            </dl>
        </div>
        <div>
            <a href="/Movies/Edit/1">Edit</a> |
            <a href="/Movies">Back to List</a>
        </div>
        ...
    </div>
    ...
</body>
</html>
```

MVC workflow example (7)

- The HTML webpage is presented by the browser

```
<div class="container body-content">
        <h2>Details</h2>
        <div>
            <h4>Movie</h4>
            <hr />
            <dl class="dl-horizontal">
                <dt> Title </dt>
                <dd> When Harry Met Sally </dd>
                <dt> Release Date </dt>
                <dd> 1989-02-12 </dd>
                <dt> Genre </dt>
                <dd> Romantic Comedy </dd>
                <dt> Price </dt>
                <dd> $7.99 </dd>
            </dl>
        </div>
        <div>
            <a href="/Movies/Edit/1">Edit</a> |
            <a href="/Movies">Back to List</a>
        </div>
        ...
    </div>
    ...
</body>
</html>
```



Details

Movie

Title	When Harry Met Sally
Release Date	1989-02-12
Genre	Romantic Comedy
Price	$7.99

Edit | Back to List

© 2020 - Movie App - Privacy

Practical

- Follow the steps at: https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-6.0&tabs=visual-studio