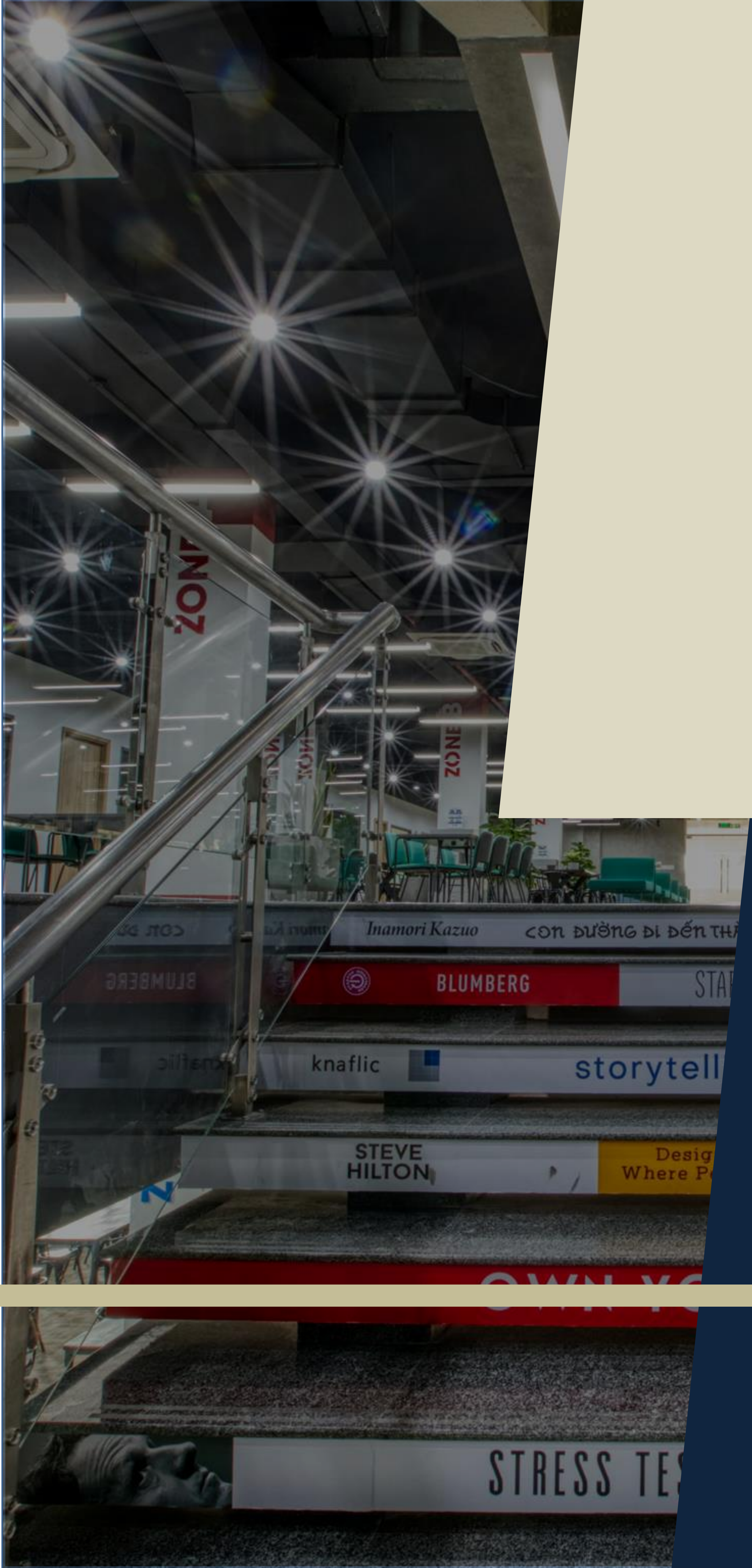


TRƯỜNG ĐẠI HỌC QUỐC TẾ MIỀN ĐÔNG
EASTERN INTERNATIONAL UNIVERSITY

# .NET Programming

## Chapter 2: ASP.NET Core MVC

## TABLE CONTENT

## .NET Framework

**Released:** 2002.

**Support Platform :** Only run in **Windows OS**.

**Mục đích:** Develop desktop application (Windows Forms, WPF), web applications (ASP.NET), web services (WCF), and develop the application for the Enterprise.

**Source Code:** Closed source.

## .NET Core

**Released**: 2016.

**Support platforms**: Multi-platform (Windows, macOS, Linux).

**Purpose**: Build modern web applications, microservices, cloud applications, and highly scalable services.

**Source code**: Open source, developed and maintained by a global community.

# .NET Framework - .NET Core

| | .NET Framework | .NET Core |
|---|---|---|
| **Operating System** | Windows only | Cross-platform: Windows, macOS, Linux |
| **Source code** | Close Source | Open-source, community contributions |
| **Performance** | Lower than .NET Core | High performance due to optimization and lightweight design |
| **Architecture** | Monolithic: Integrates many features into a large application | Modular: Uses NuGet packages, includes only necessary components |
| **Dependency Injection** | Supported through external libraries like Unity or Ninject | Built-in, powerful DI system |
| **Middleware Pipeline** | Limited customization of the request processing pipeline | Flexible middleware pipeline, easy to customize |
| **Container Support** | Little support, not optimized for containerization | Optimized for containers, easy to deploy on Docker and Kubernetes |
| **Configuration** | Uses complex Web.config file | Simpler configuration through appsettings.json file and flexible code configuration |
| **Security** | Integrated with Windows security features like Active Directory | Provides modern security features like OAuth, JWT, easy to integrate with external security services |
| **Framework Support** | Limited, heavily dependent on .NET Framework | Good support for many frameworks and new technologies like Blazor, gRPC |
| **Updates and Support** | Slow updates, mainly focused on maintaining legacy applications | Frequent updates, receives new features quickly |
| **Cloud Deployment** | Not optimized for cloud | Optimized for cloud deployment |

## WHAT IS ASP.NET ?

ASP.NET is a robust web application development framework created by Microsoft. It empowers developers to build dynamic web applications, web services, and APIs using programming languages such as C# or VB.NET. ASP.NET supports various development models, enabling the creation of efficient, secure, and maintainable web applications.

## ASP.NET

**Release**: Launched in 2002 as part of the .NET Framework.

**Supported platforms**: Runs only on the Windows operating system.

**Architecture**: Built on the .NET Framework with models such as Web Forms, MVC (from ASP.NET MVC 1.0 onwards), and Web API.

## ASP.NET Core

**Release**: Launched in 2016 as part of .NET Core, later becoming part of the .NET platform from .NET 5 onwards.

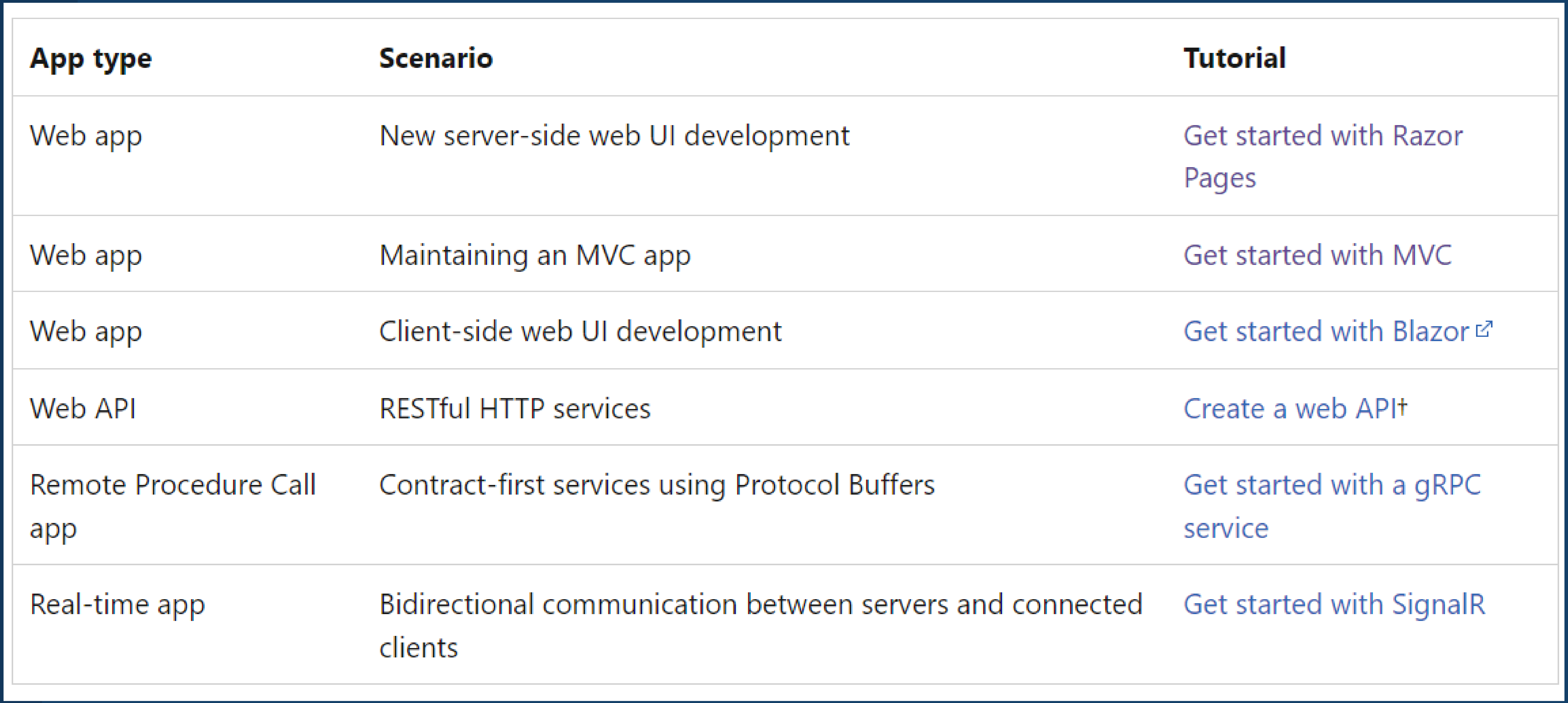
**Supported platforms**: Cross-platform (Windows, macOS, Linux).

**Architecture**: Modular, lightweight, and performance-optimized design, supporting models such as MVC, Razor Pages, Blazor, and Web API.

## ASP.NET - ASP.NET Core

| | ASP.NET | ASP.NET Core |
|---|---|---|
| Supported Platforms | Windows only | Cross-platform: Windows, macOS, Linux |
| Source Code | Primarily closed-source | Open-source, community-driven |
| Performance | Lower performance compared to ASP.NET Core | High performance due to optimization and lightweight design |
| Architecture | Monolithic: Integrates many features into a large application | Modular: Uses NuGet packages, includes only necessary components |
| Middleware | Limited customization of the request processing pipeline | Flexible middleware pipeline, easy to customize |
| Dependency Injection | Supported through external libraries | Built-in, powerful DI system |
| Container Support | Limited support, not optimized for containerization | Optimized for containers, easy to deploy on Docker and Kubernetes |
| Configuration and Deployment | Uses complex Web.config file | Simpler configuration through appsettings.json file and flexible code-based configuration |
| Security | Integrated with Windows security features | Provides modern security features, supports OAuth, JWT, etc. |
| Framework Support | Limited, heavily dependent on .NET Framework | Supports a wide range of frameworks and emerging technologies like Blazor, gRPC |
| Updates and Support | Slow updates, primarily focused on maintenance | Frequent updates, receives new features quickly |

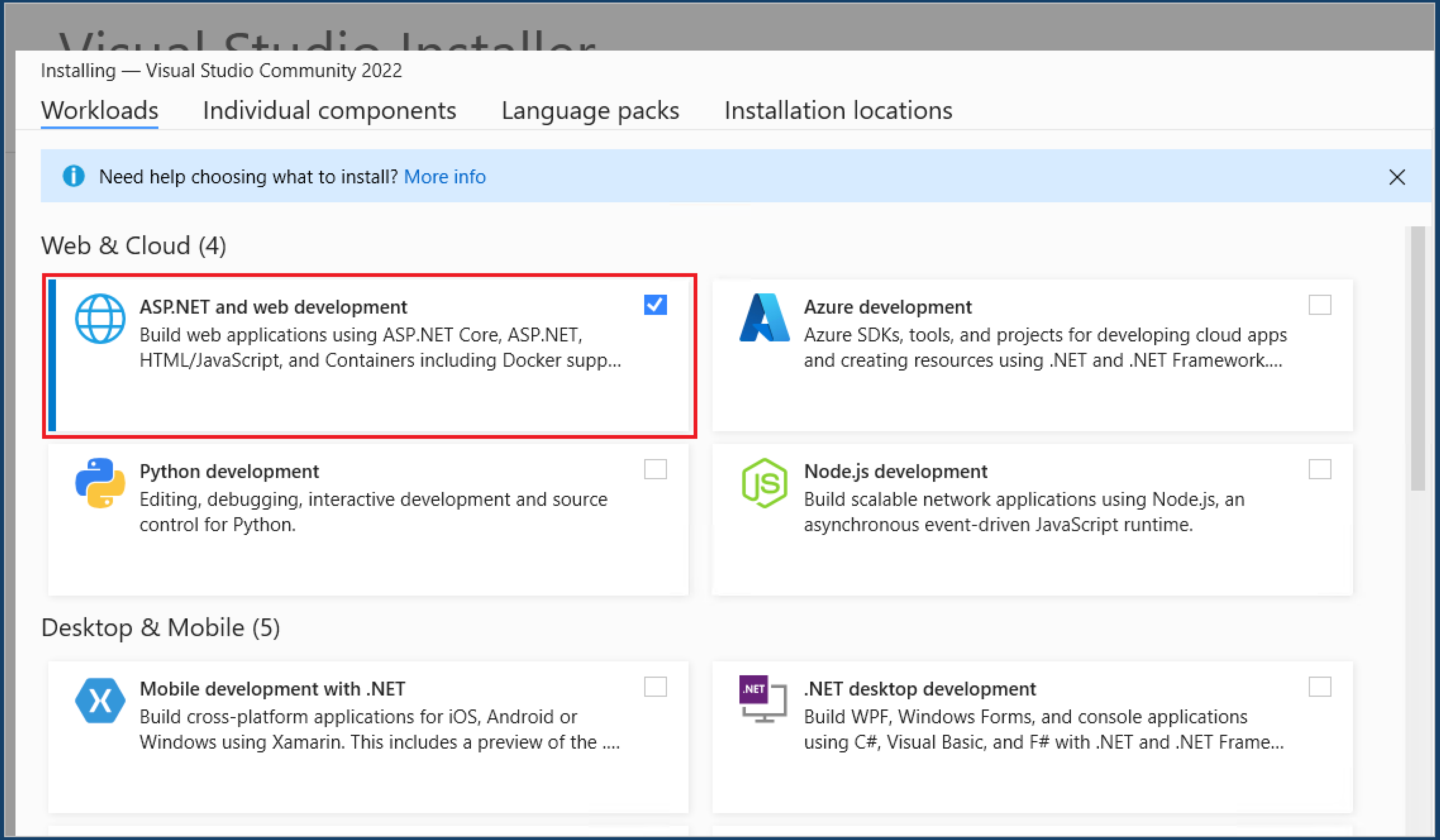
# Get started with ASP.NET Core



| App type | Scenario | Tutorial |
|---|---|---|
| Web app | New server-side web UI development | Get started with Razor Pages |
| Web app | Maintaining an MVC app | Get started with MVC |
| Web app | Client-side web UI development | Get started with Blazor |
| Web API | RESTful HTTP services | Create a web API† |
| Remote Procedure Call app | Contract-first services using Protocol Buffers | Get started with a gRPC service |
| Real-time app | Bidirectional communication between servers and connected clients | Get started with SignalR |

# ASP.NET Core MVC

Prerequisites: .NET Core 8.0



Installing — Visual Studio Community 2022

Need help choosing what to install? More info

Web & Cloud (4)

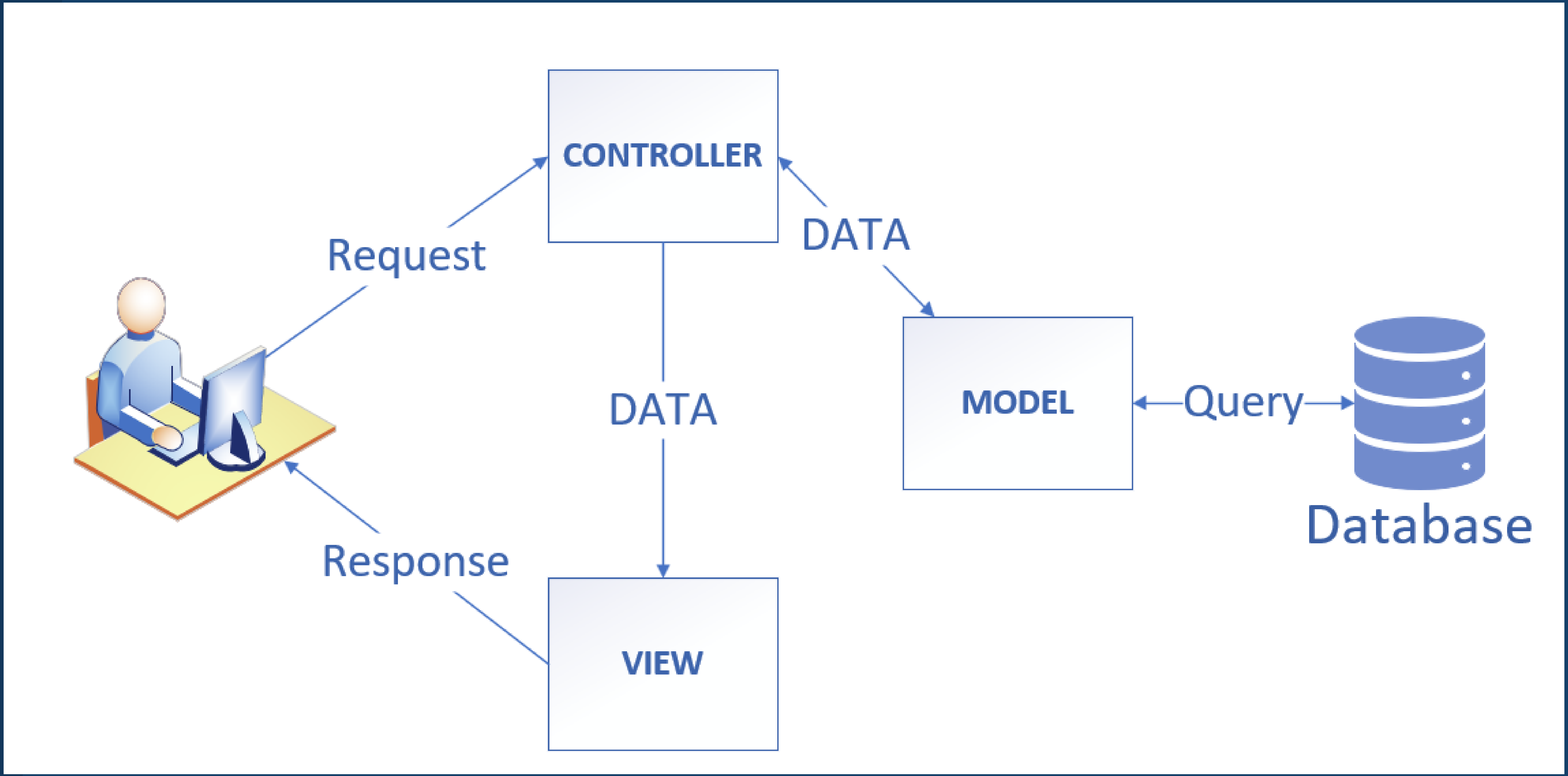
- [✓] **ASP.NET and web development** Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...
- [ ] **Azure development** Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET and .NET Framework....
- [ ] **Python development** Editing, debugging, interactive development and source control for Python.
- [ ] **Node.js development** Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

- [ ] **Mobile development with .NET** Build cross-platform applications for iOS, Android or Windows using Xamarin. This includes a preview of the ....
- [ ] **.NET desktop development** Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...

## ASP.NET Core MVC

MVC Architecture:



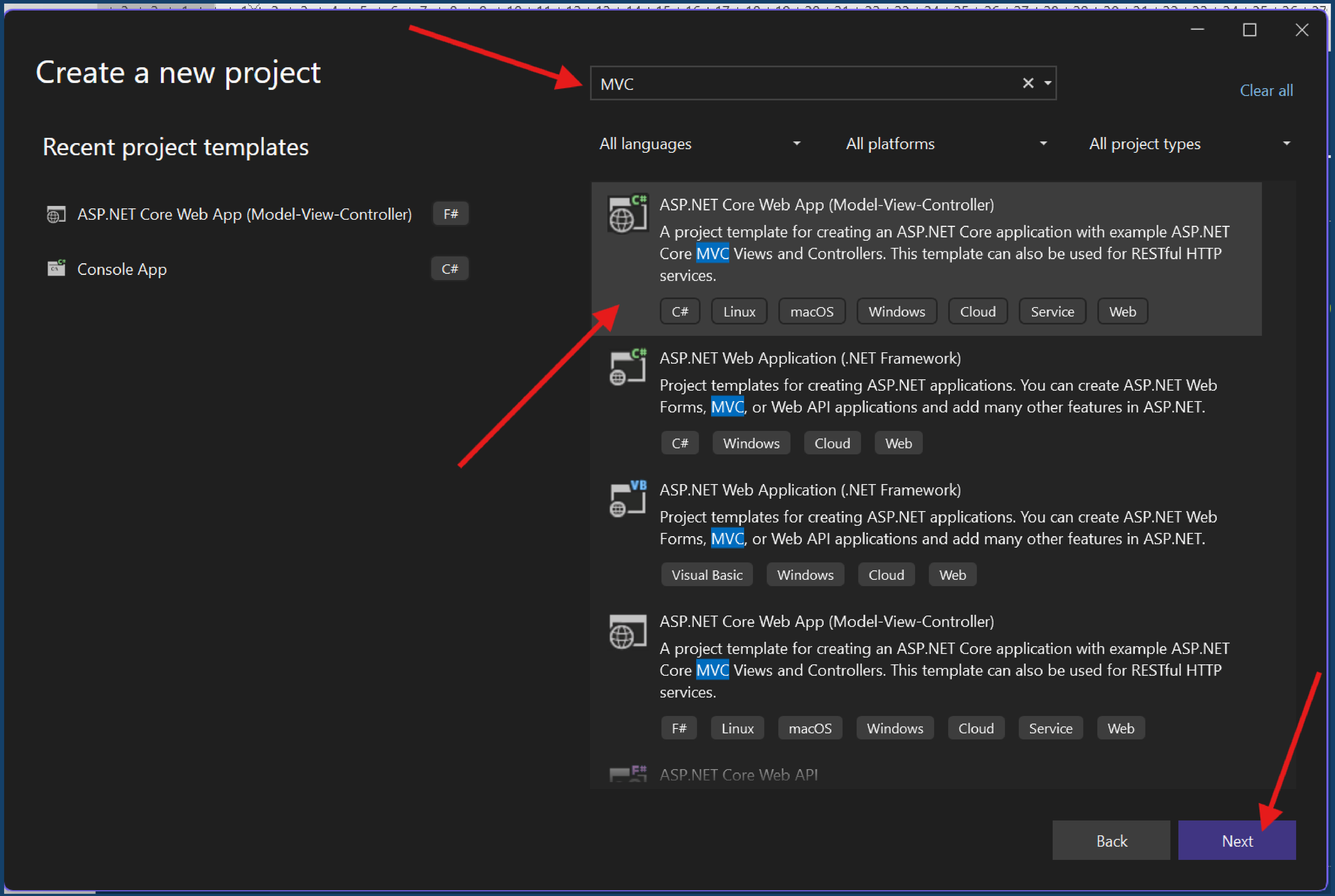
## ASP.NET Core MVC

**Create a web app:**

- Start Visual Studio and select **Create a new project**.
- In the **Create a new project** dialog, select **ASP.NET Core Web App (Model-View-Controller)** **> Next.**
- In the **Configure your new project** dialog:
  - Enter LibraryManagement for Project name. It's important to name the project LibraryManagement. Capitalization needs to match each namespace when code is copied.
  - The Location for the project can be set to anywhere.
- Select Next.
- In the Additional information dialog:
  - Select .NET 8.0 (Long Term Support).
  - Verify that Do not use top-level statements is unchecked.
- Select Create.

# ASP.NET Core MVC

## Create a web app:



Create a new project

MVC

Clear all

Recent project templates

ASP.NET Core Web App (Model-View-Controller) F#

Console App C#

All languages | All platforms | All project types

ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
C# Linux macOS Windows Cloud Service Web

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
C# Windows Cloud Web

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
Visual Basic Windows Cloud Web

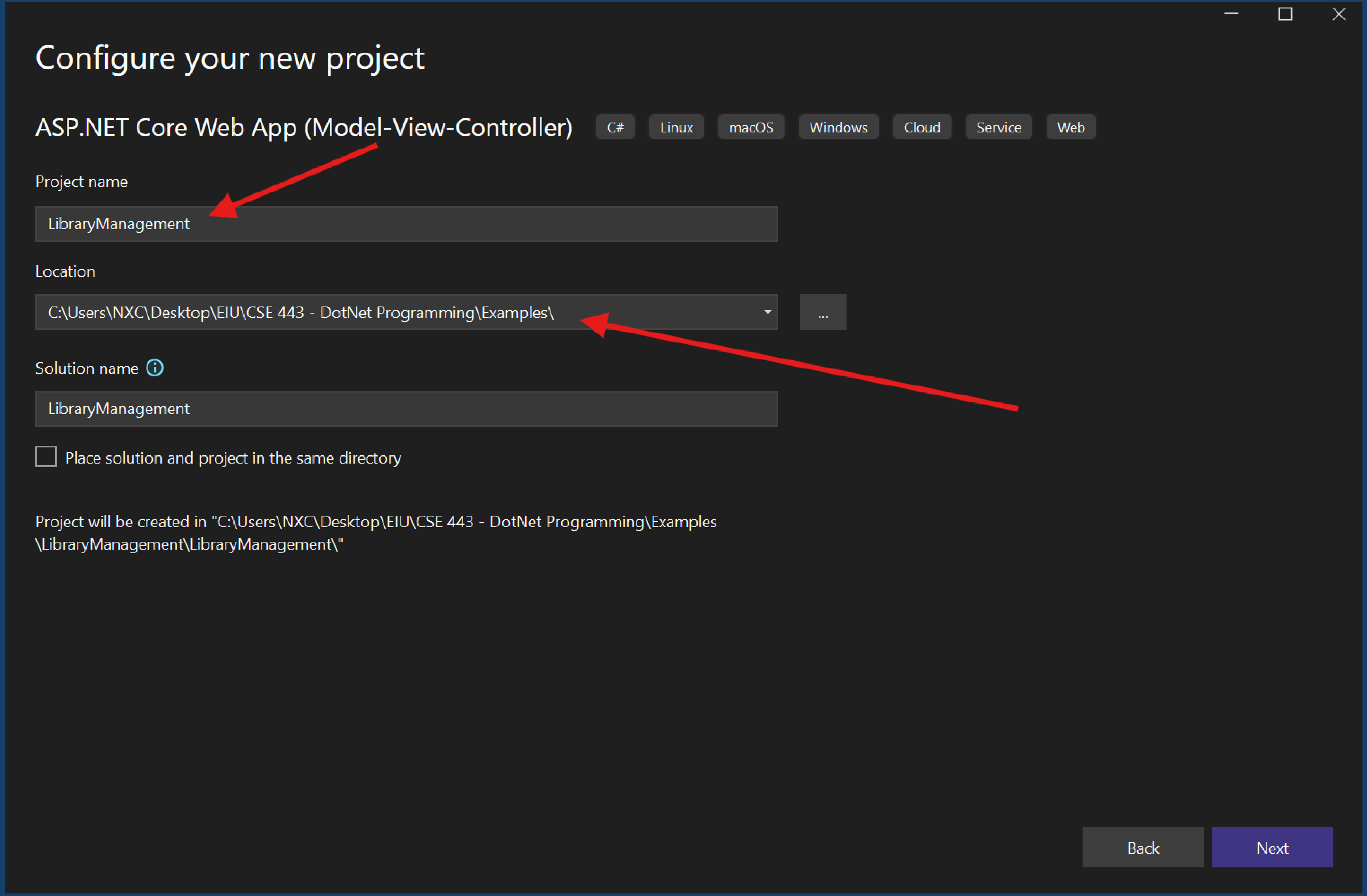
ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
F# Linux macOS Windows Cloud Service Web

ASP.NET Core Web API

Back Next

## ASP.NET Core MVC

### Create a web app:



Configure your new project

ASP.NET Core Web App (Model-View-Controller)

Project name: LibraryManagement

Location: C:\Users\NXC\Desktop\EIU\CSE 443 - DotNet Programming\Examples\

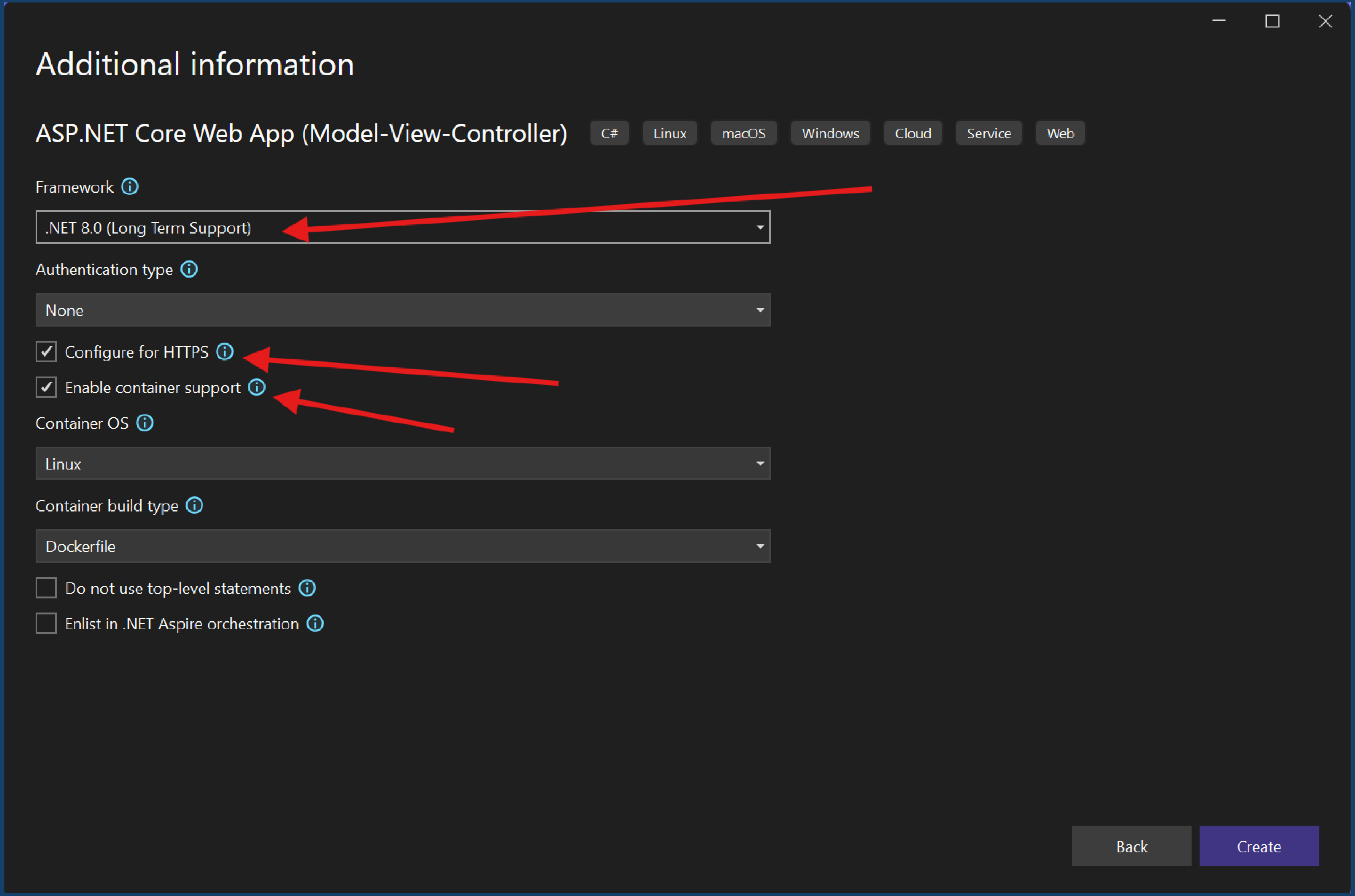
Solution name: LibraryManagement

☐ Place solution and project in the same directory

Project will be created in "C:\Users\NXC\Desktop\EIU\CSE 443 - DotNet Programming\Examples\LibraryManagement\LibraryManagement\"

# ASP.NET Core MVC

## Create a web app:



Additional information

ASP.NET Core Web App (Model-View-Controller)

Framework: .NET 8.0 (Long Term Support)

Authentication type: None

☑ Configure for HTTPS

☑ Enable container support

Container OS: Linux

Container build type: Dockerfile

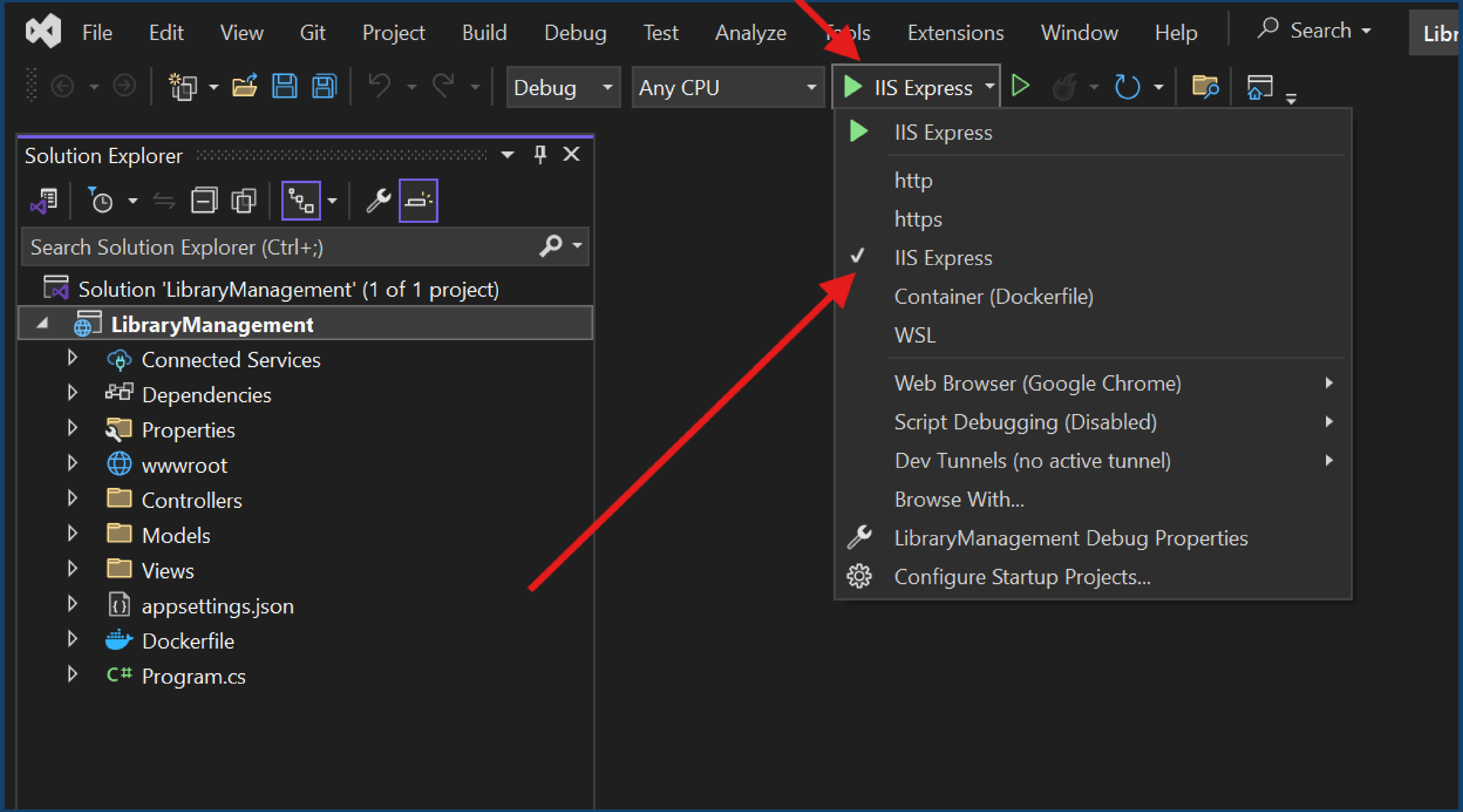
☐ Do not use top-level statements

☐ Enlist in .NET Aspire orchestration

EIU

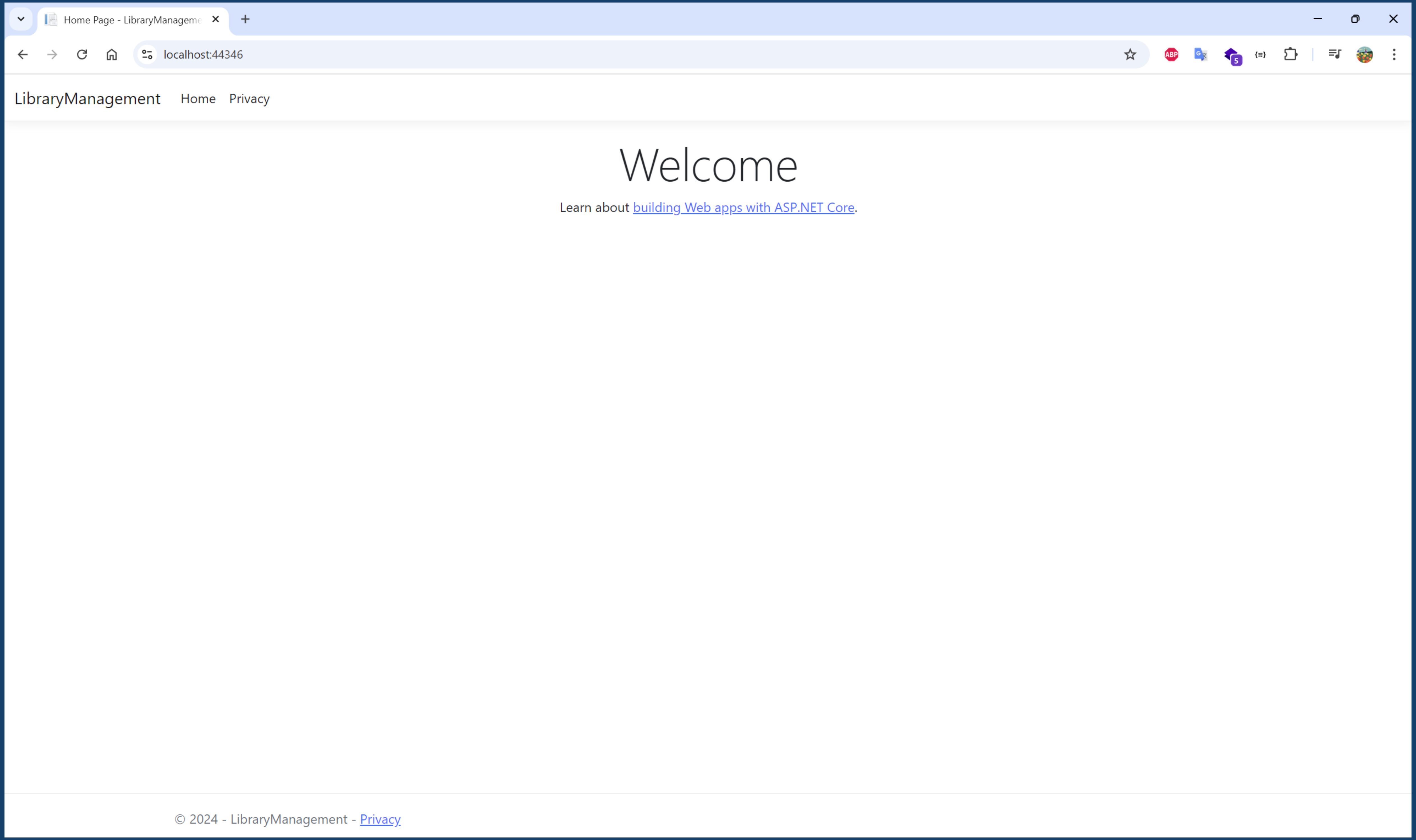
# ASP.NET Core MVC

## Run the app



# ASP.NET Core MVC

## Run the app



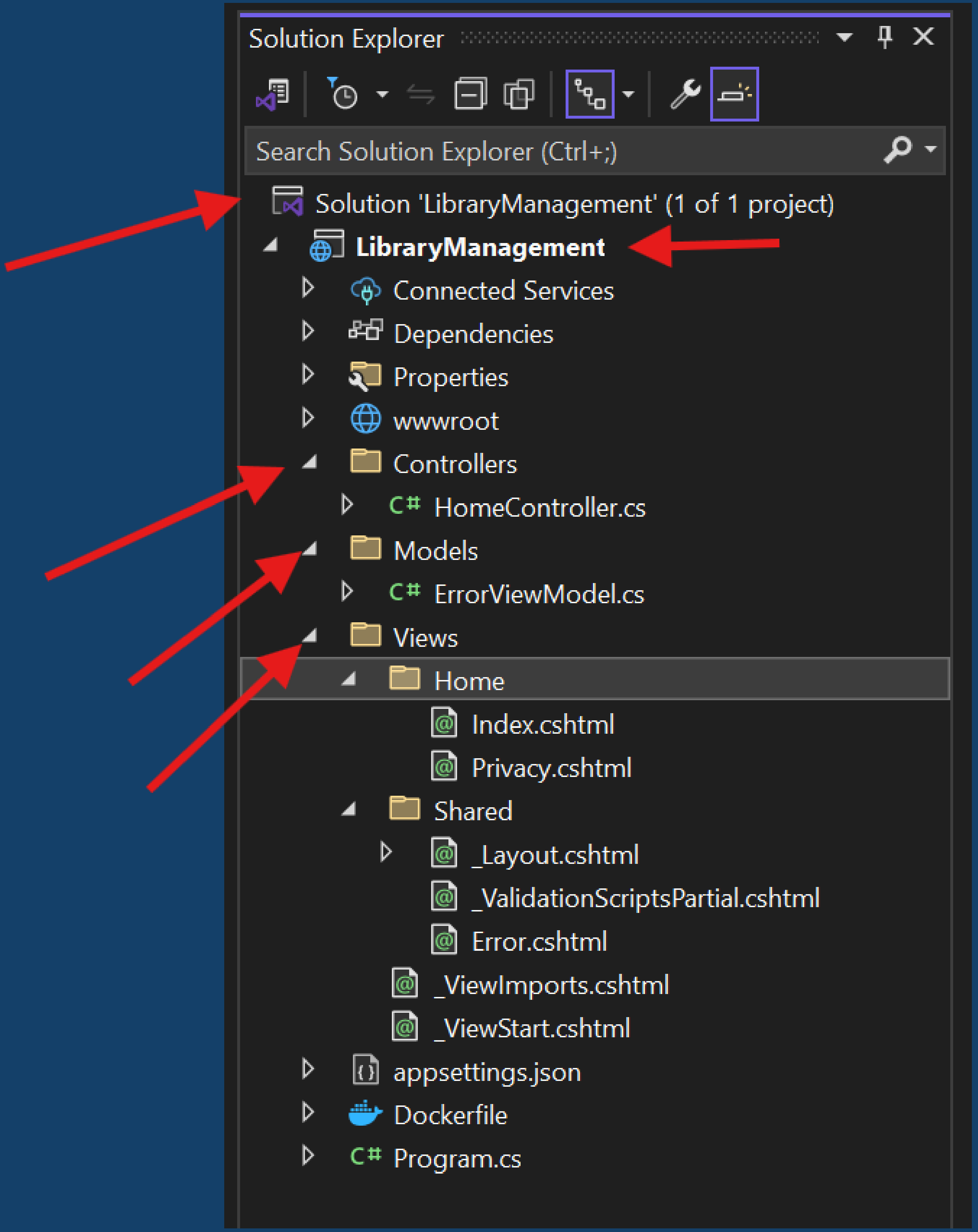
Welcome

Learn about building Web apps with ASP.NET Core.

© 2024 - LibraryManagement - Privacy

# ASP.NET Core MVC

## Folder struct



```
Solution 'LibraryManagement' (1 of 1 project)
  LibraryManagement
    Connected Services
    Dependencies
    Properties
    wwwroot
    Controllers
      HomeController.cs
    Models
      ErrorViewModel.cs
    Views
      Home
        Index.cshtml
        Privacy.cshtml
      Shared
        _Layout.cshtml
        _ValidationScriptsPartial.cshtml
        Error.cshtml
      _ViewImports.cshtml
      _ViewStart.cshtml
    appsettings.json
    Dockerfile
    Program.cs
```

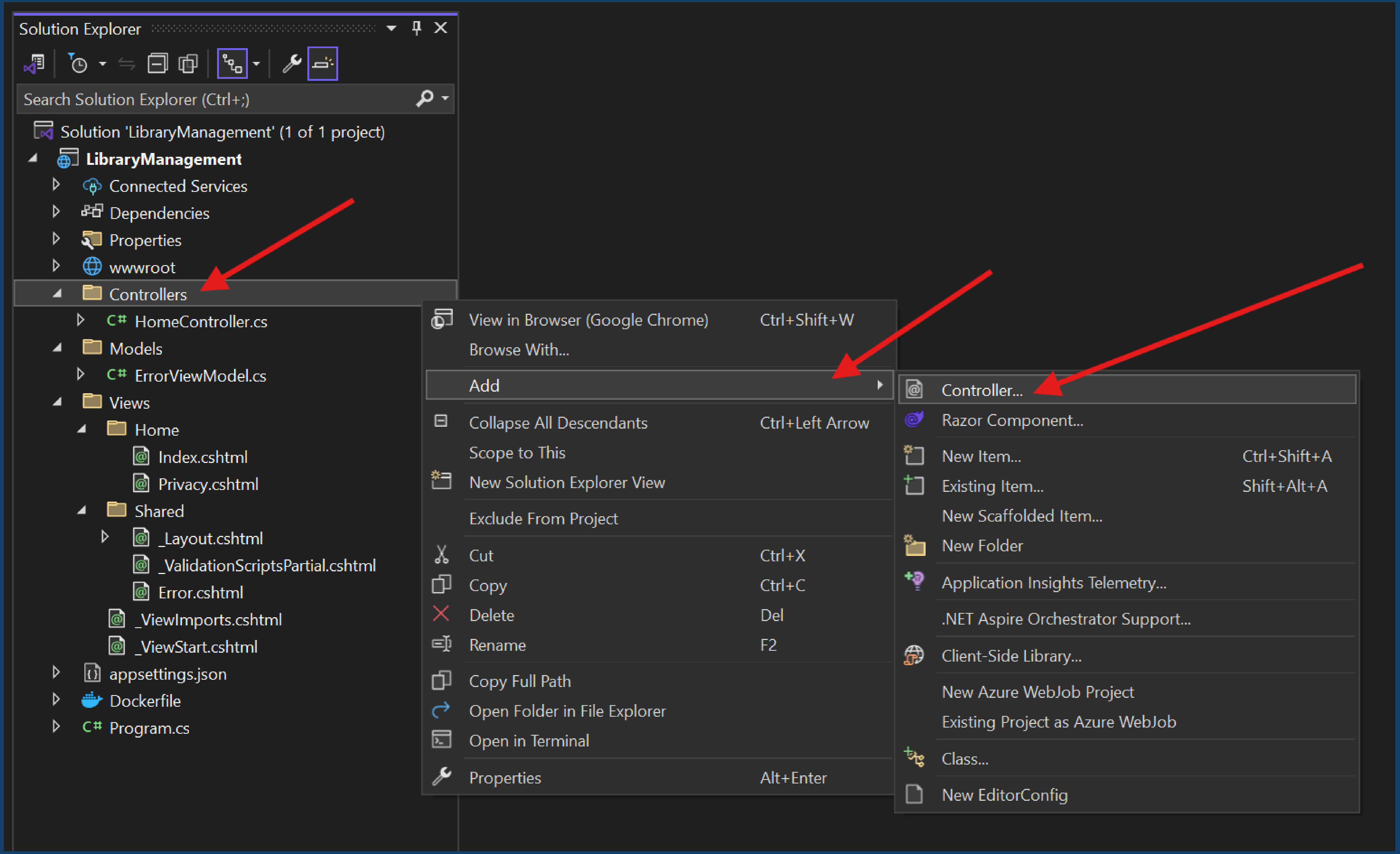
## ASP.NET Core MVC

**Controller**

- Handles user input and interactions.
- Processes requests, retrieves model data, and selects the appropriate view for response.
- Example: Handles URL requests like => localhost:44346/Home/Privacy.

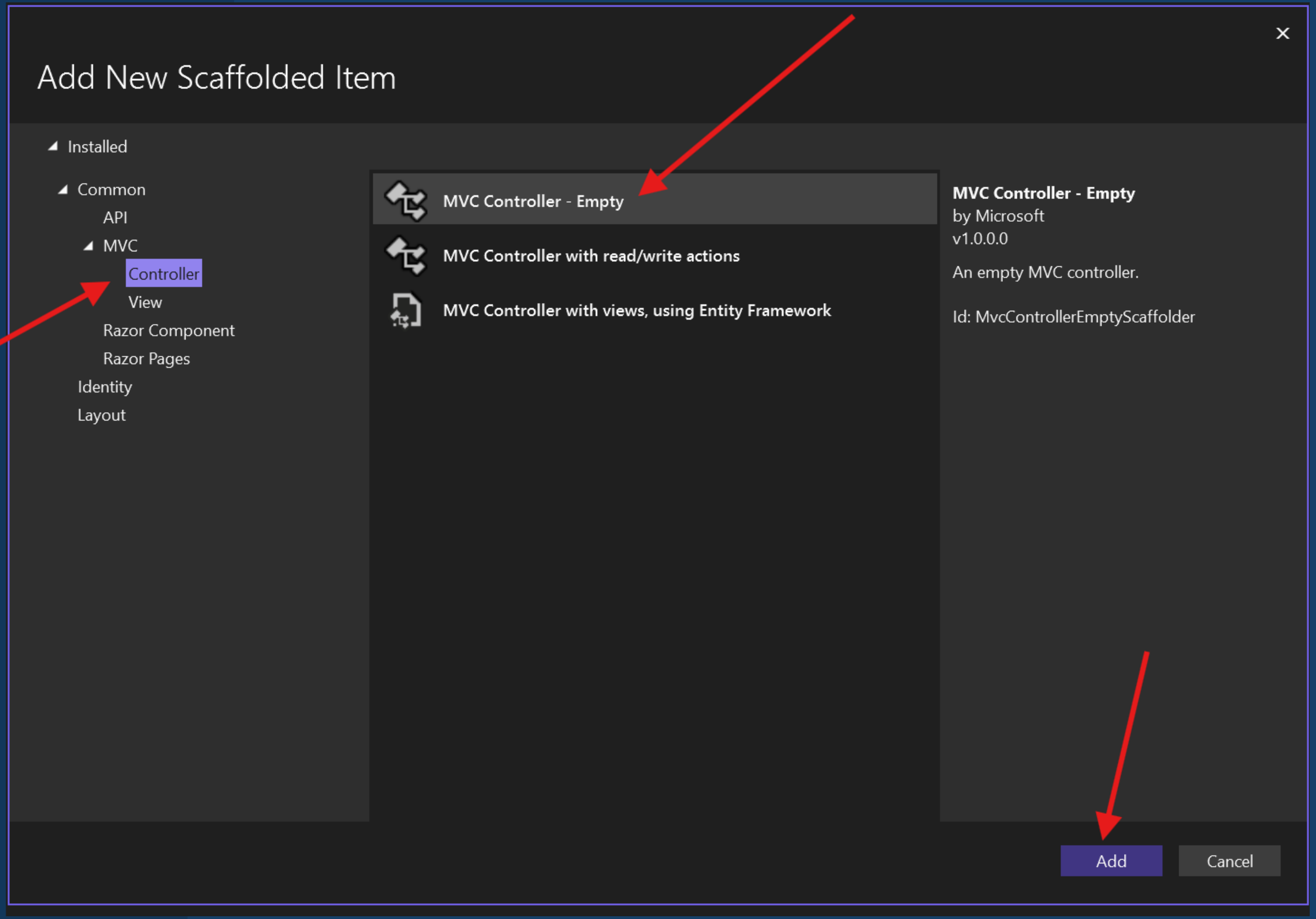
# ASP.NET Core MVC

## Controller - Add a controller

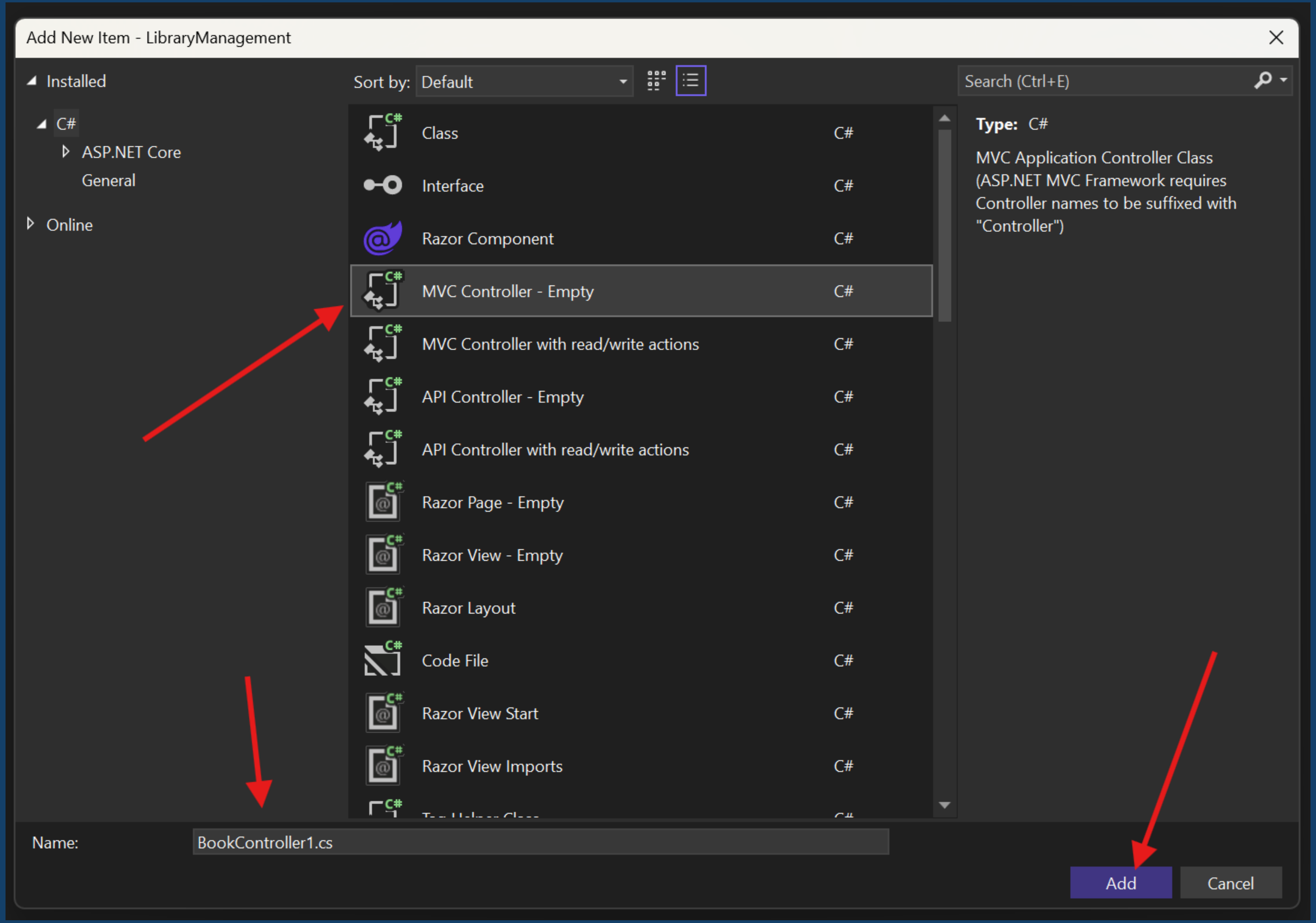


# ASP.NET Core MVC

## Controller - Add a controller



MVC Controller - Empty
by Microsoft
v1.0.0.0
An empty MVC controller.
Id: MvcControllerEmptyScaffolder



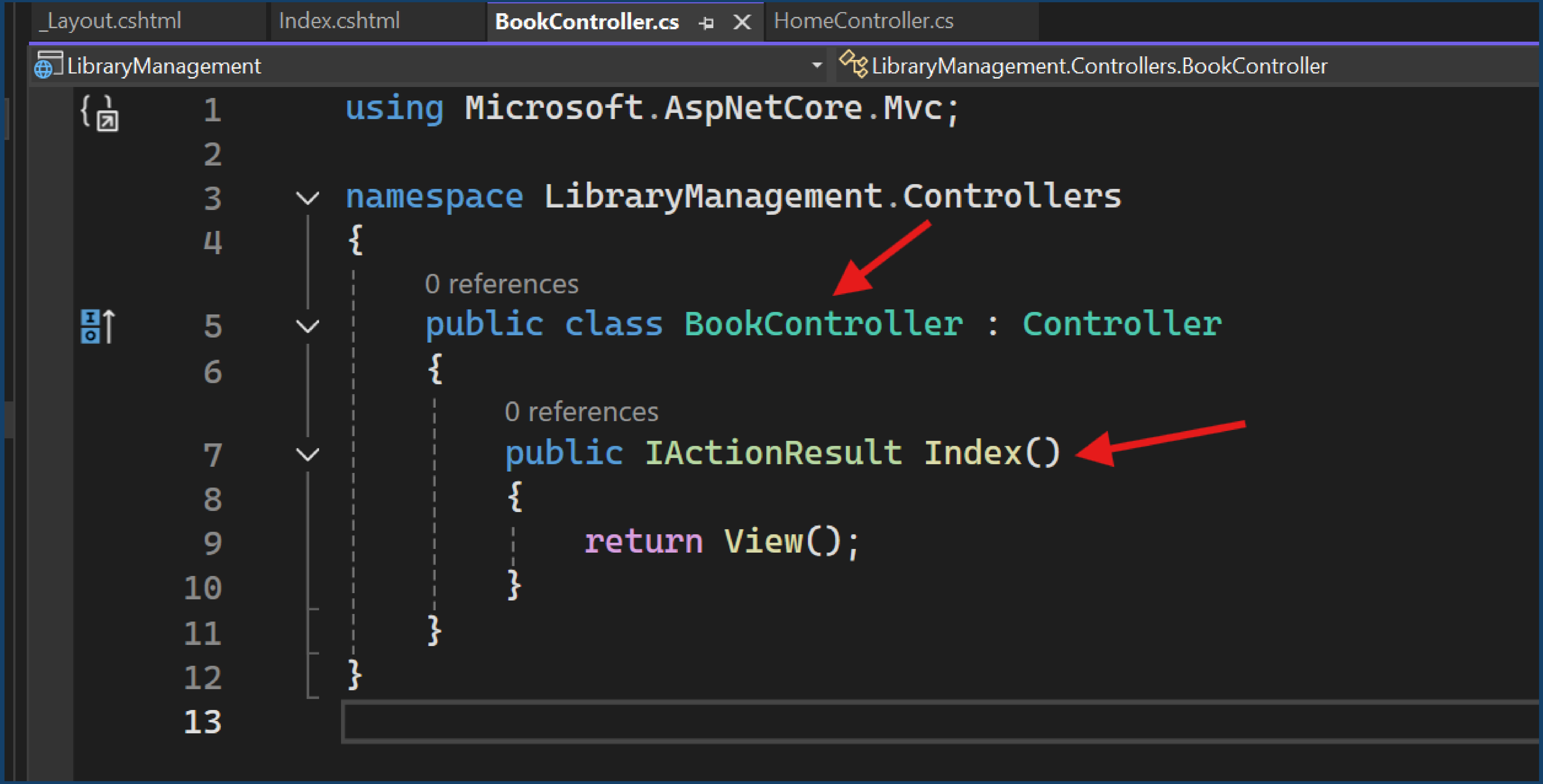
Type: C#

MVC Application Controller Class (ASP.NET MVC Framework requires Controller names to be suffixed with "Controller")

Name: BookController1.cs

# ASP.NET Core MVC

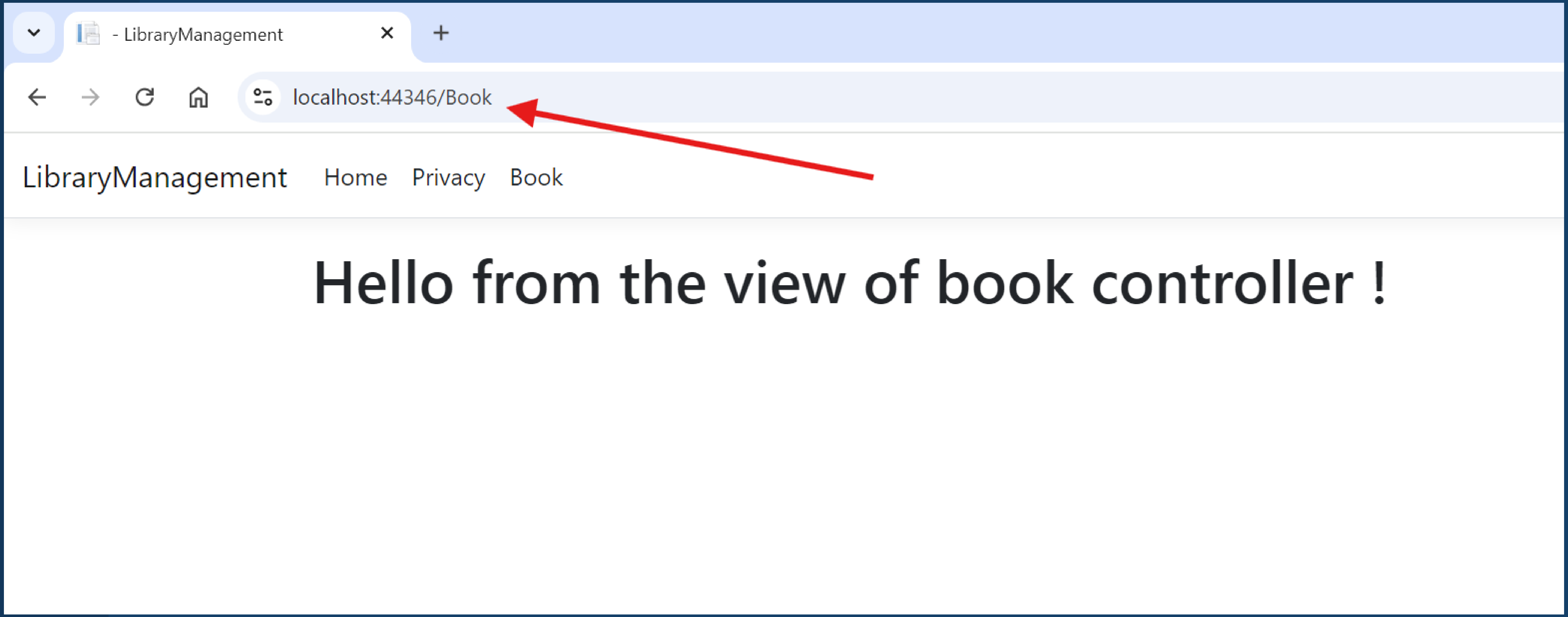
## Controller – Index method



```csharp
using Microsoft.AspNetCore.Mvc;

namespace LibraryManagement.Controllers
{
    0 references
    public class BookController : Controller
    {
        0 references
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

## ASP.NET Core MVC

### Controller – view from controller



- LibraryManagement

localhost:44346/Book

LibraryManagement Home Privacy Book

Hello from the view of book controller !

# ASP.NET Core MVC

## Controller – HTTP Endpoint

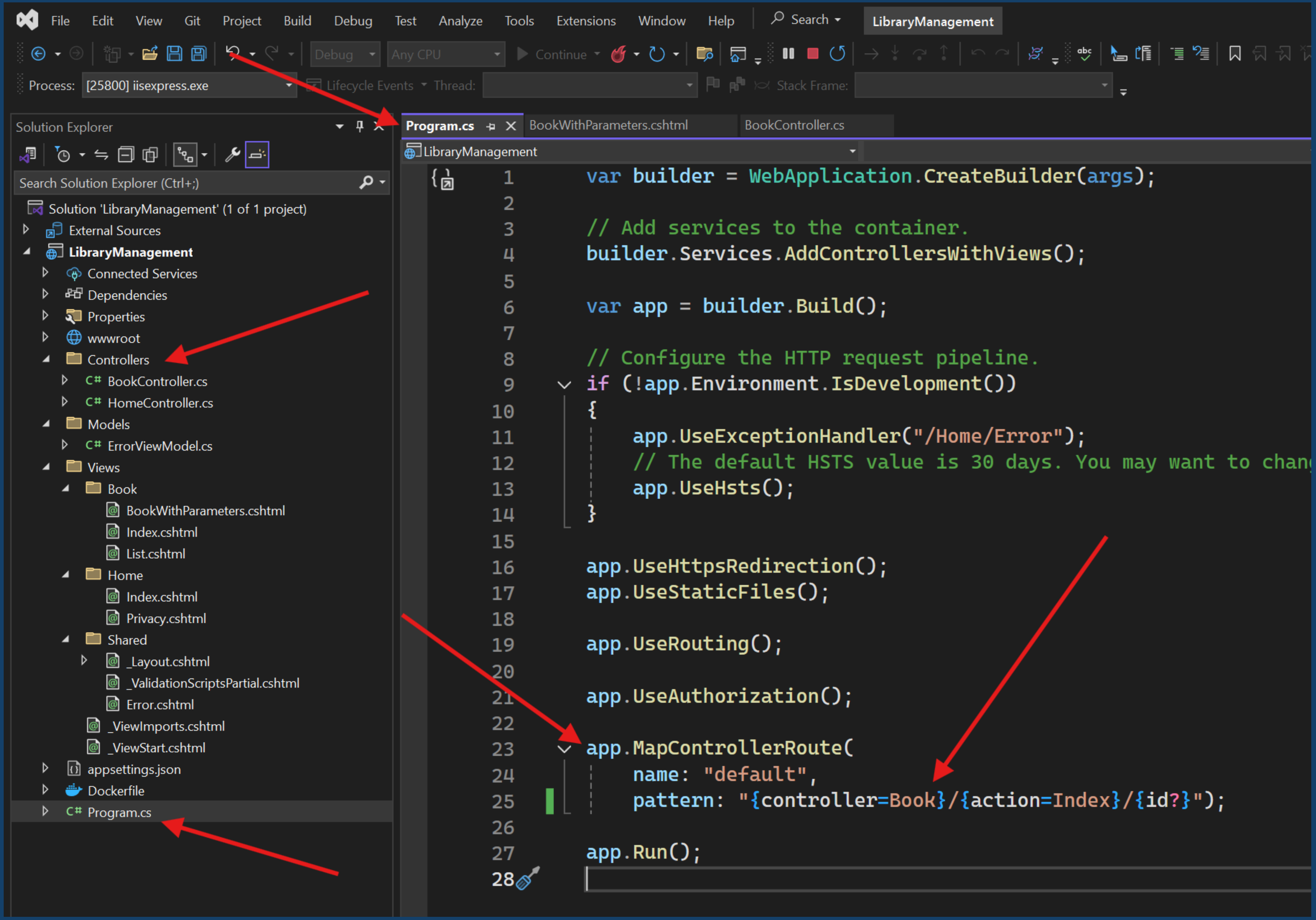
Every public method in a controller is callable as an HTTP endpoint
An HTTP endpoint:
Is a targetable URL in the web application, such as https://localhost:44346/Book.
Combines:

- The protocol used: HTTPS.
- The network location of the web server, including the TCP port: localhost:44346.
- The target URI: HelloWorld.

# ASP.NET Core MVC

## Controller – Default Endpoint

Every time a user accesses the website domain, the default path will be set in Program.cs



Solution 'LibraryManagement' (1 of 1 project)
- External Sources
- LibraryManagement
  - Connected Services
  - Dependencies
  - Properties
  - wwwroot
  - Controllers
    - BookController.cs
    - HomeController.cs
  - Models
    - ErrorViewModel.cs
  - Views
    - Book
      - BookWithParameters.cshtml
      - Index.cshtml
      - List.cshtml
    - Home
      - Index.cshtml
      - Privacy.cshtml
    - Shared
      - _Layout.cshtml
      - _ValidationScriptsPartial.cshtml
      - Error.cshtml
    - _ViewImports.cshtml
    - _ViewStart.cshtml
  - appsettings.json
  - Dockerfile
  - Program.cs

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to chang
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Book}/{action=Index}/{id?}");

app.Run();
```

## ASP.NET Core MVC

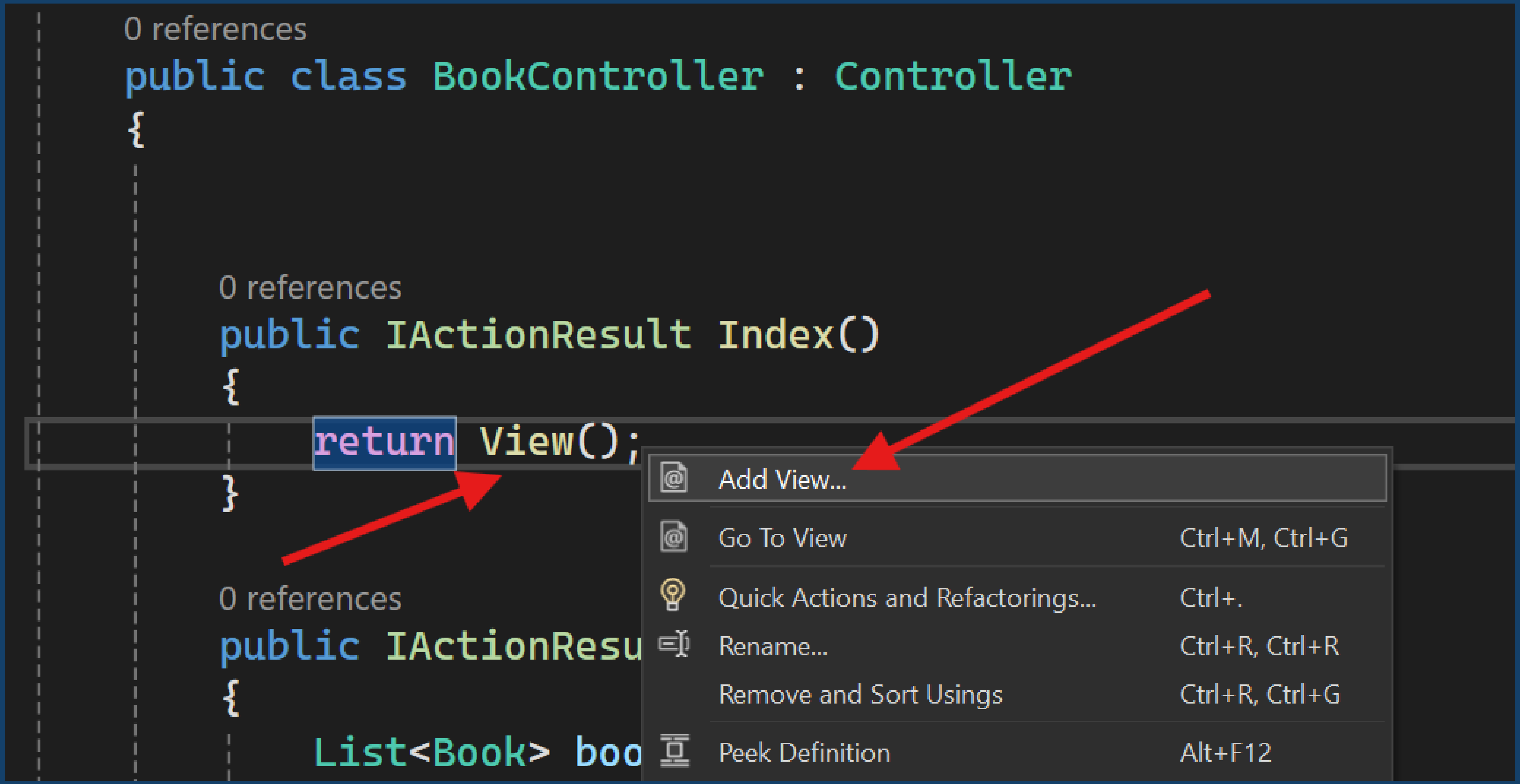
# View

## ASP.NET Core MVC

### View - add a view to an ASP.NET Core MVC app

View templates are created using Razor. Razor-based view templates:

- Have a .cshtml file extension.
- Provide an elegant way to create HTML output with C#.

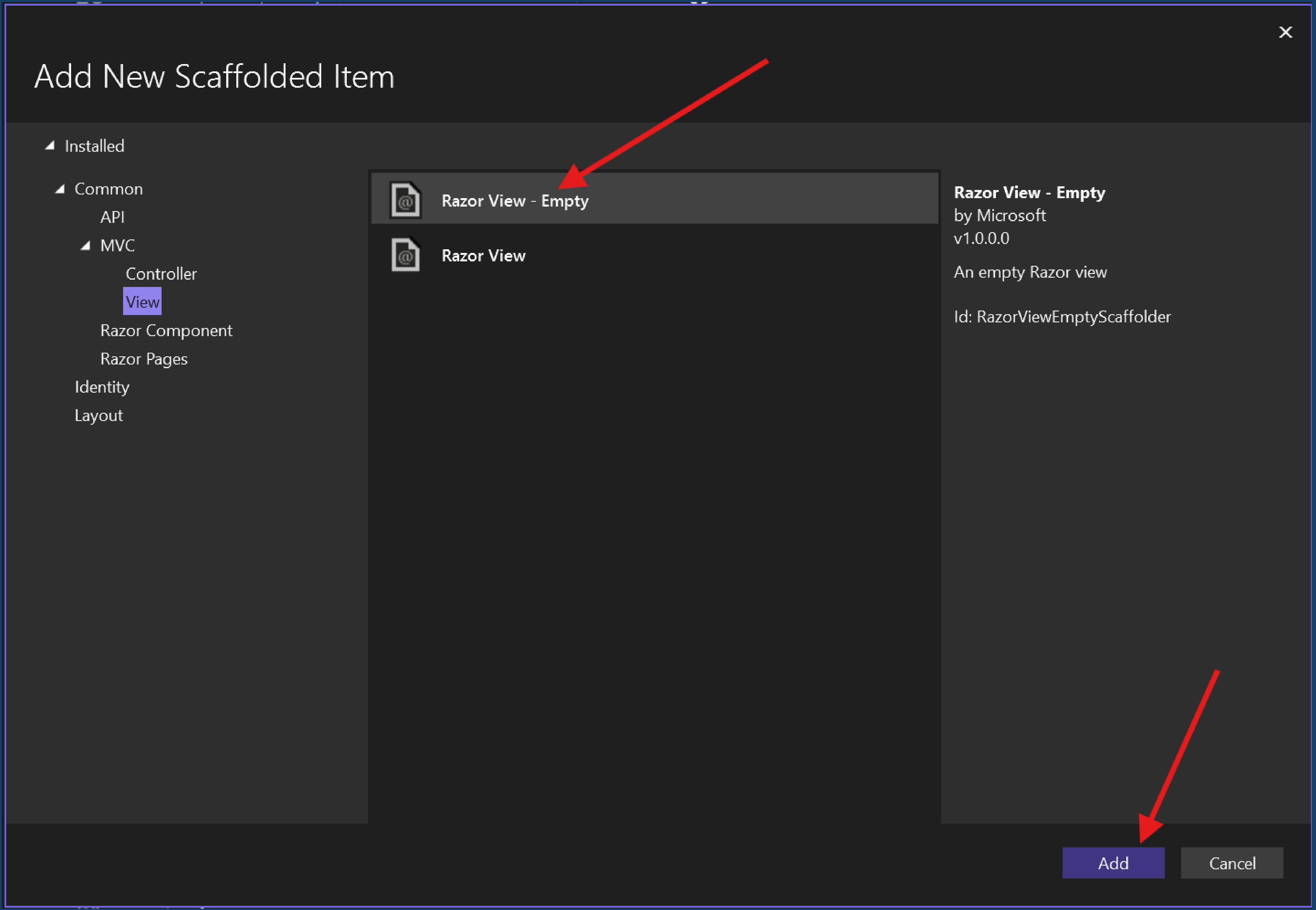


```
0 references
public class BookController : Controller
{


    0 references
    public IActionResult Index()
    {
        return View();
    }

    0 references
    public IActionResu
    {
        List<Book> boo
```

Add View...
Go To View Ctrl+M, Ctrl+G
Quick Actions and Refactorings... Ctrl+.
Rename... Ctrl+R, Ctrl+R
Remove and Sort Usings Ctrl+R, Ctrl+G
Peek Definition Alt+F12

# ASP.NET Core MVC

## View - add a view to an ASP.NET Core MVC app



Add New Scaffolded Item

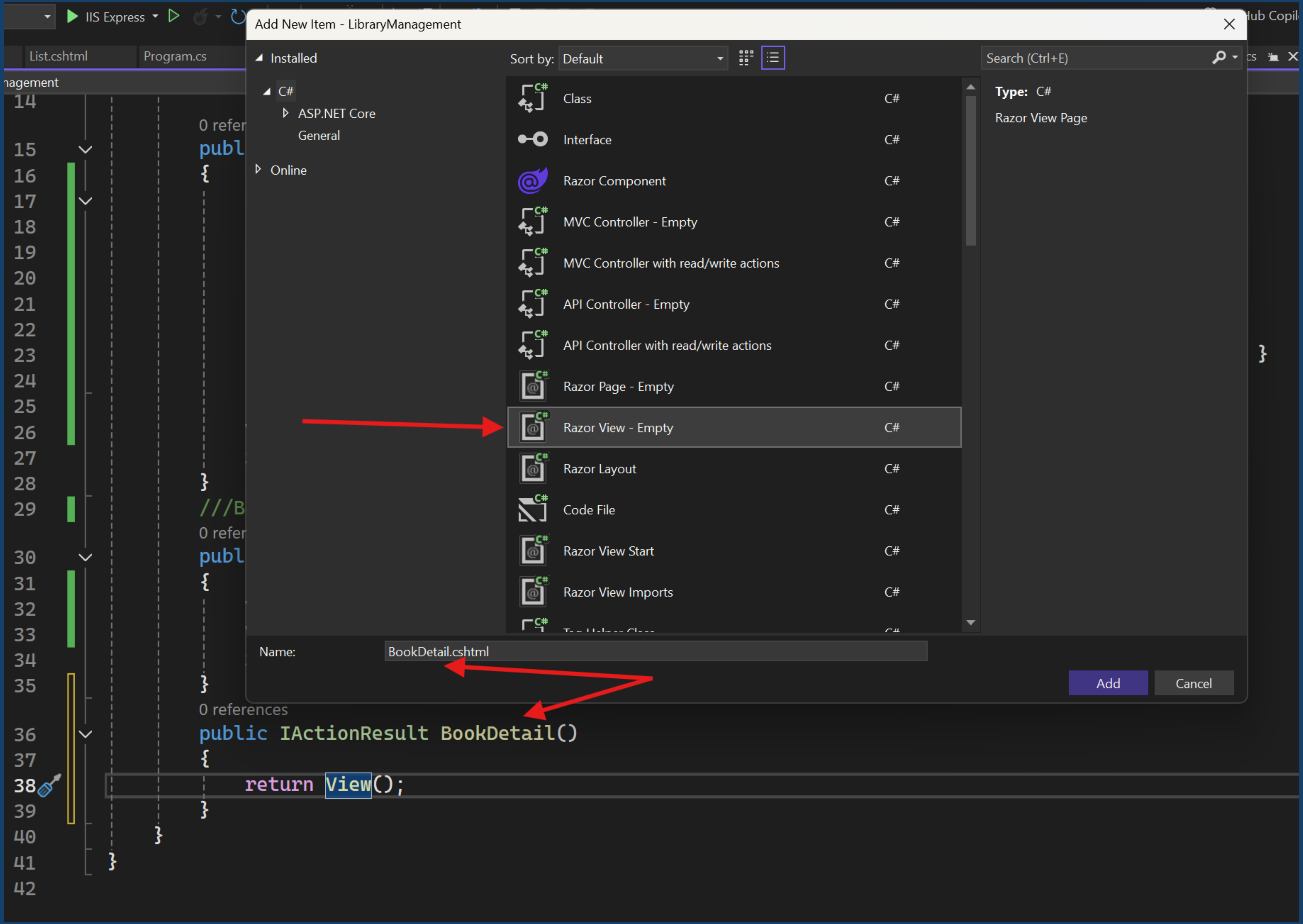
**Razor View - Empty**
by Microsoft
v1.0.0.0

An empty Razor view

Id: RazorViewEmptyScaffolder

# ASP.NET Core MVC

## View - add a view to an ASP.NET Core MVC app



Add New Item - LibraryManagement

Installed

- C#
  - ASP.NET Core
  - General
- Online

Sort by: Default

| | |
|---|---|
| Class | C# |
| Interface | C# |
| Razor Component | C# |
| MVC Controller - Empty | C# |
| MVC Controller with read/write actions | C# |
| API Controller - Empty | C# |
| API Controller with read/write actions | C# |
| Razor Page - Empty | C# |
| Razor View - Empty | C# |
| Razor Layout | C# |
| Code File | C# |
| Razor View Start | C# |
| Razor View Imports | C# |

Search (Ctrl+E)

**Type:** C#

Razor View Page

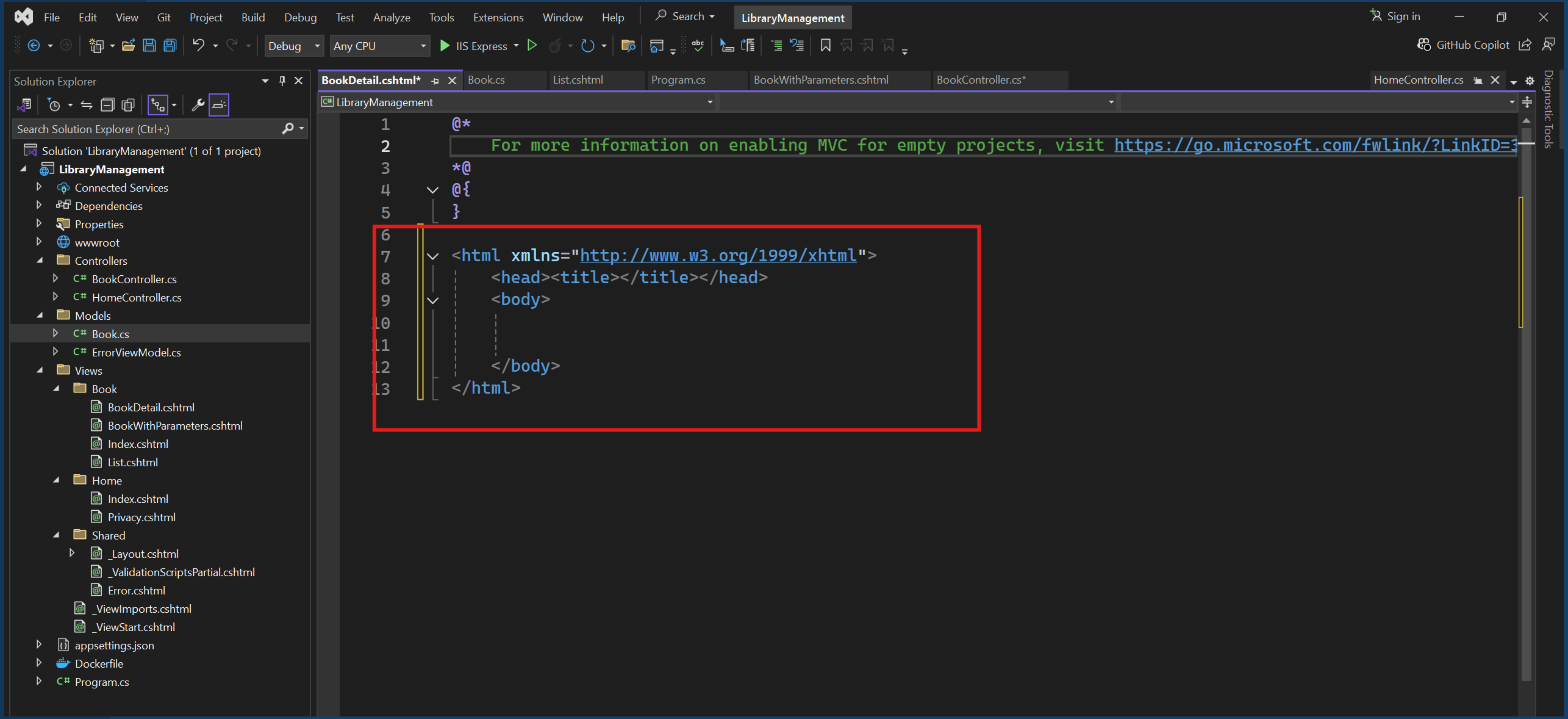
Name: BookDetail.cshtml

Add Cancel

```
        0 references
        public IActionResult BookDetail()
        {
            return View();
        }
    }
}
```

# ASP.NET Core MVC

## View – The Razor view after create success

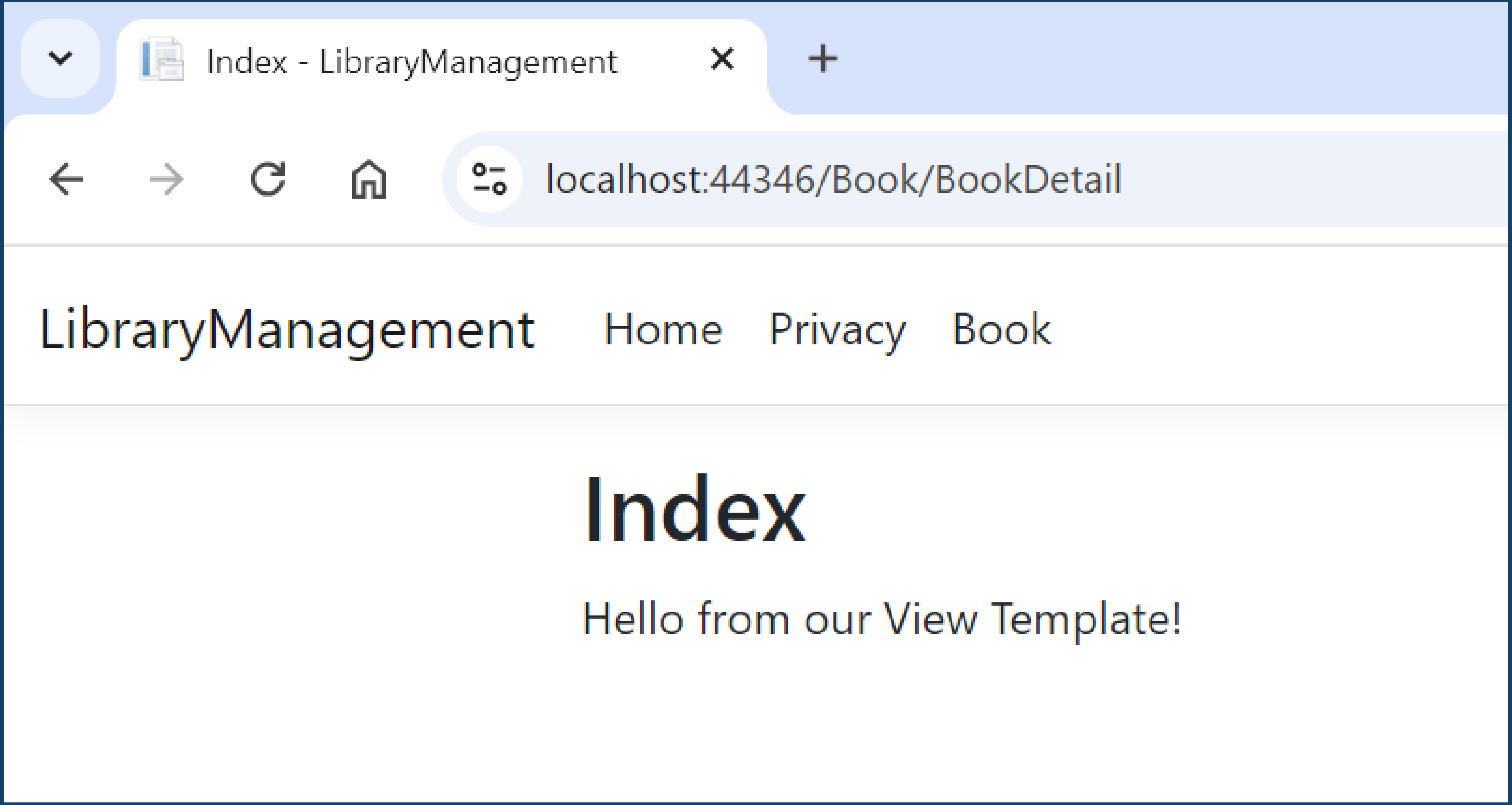


```
Solution 'LibraryManagement' (1 of 1 project)
  LibraryManagement
    Connected Services
    Dependencies
    Properties
    wwwroot
    Controllers
      BookController.cs
      HomeController.cs
    Models
      Book.cs
      ErrorViewModel.cs
    Views
      Book
        BookDetail.cshtml
        BookWithParameters.cshtml
        Index.cshtml
        List.cshtml
      Home
        Index.cshtml
        Privacy.cshtml
      Shared
        _Layout.cshtml
        _ValidationScriptsPartial.cshtml
        Error.cshtml
      _ViewImports.cshtml
      _ViewStart.cshtml
    appsettings.json
    Dockerfile
    Program.cs
```

```
@*
    For more information on enabling MVC for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=3
*@
@{
}

<html xmlns="http://www.w3.org/1999/xhtml">
    <head><title></title></head>
    <body>


    </body>
</html>
```

## ASP.NET Core MVC

**View – Show in the website**
**Navigate to https://localhost:{PORT}/{Controller}: // check the path in your project**


Index - LibraryManagement

localhost:44346/Book/BookDetail

Index

Hello from our View Template!

## ASP.NET Core MVC

**View – Change views and layout pages**

**Select the menu links LibraryManagement, Home, and Privacy. Each page shows the same menu layout. The menu layout is implemented in the Views/Shared/_Layout.cshtml file.**

**Open the Views/Shared/_Layout.cshtml file.**

**Layout templates allow:**

**- Specifying the HTML container layout of a site in one place.**

**- Applying the HTML container layout across multiple pages in the site.**

## ASP.NET Core MVC

**View – Change views and layout pages**

**Select the menu links LibraryManagement, Home, and Privacy. Each page shows the same menu layout. The menu layout is implemented in the Views/Shared/_Layout.cshtml file.**

**Open the Views/Shared/_Layout.cshtml file.**

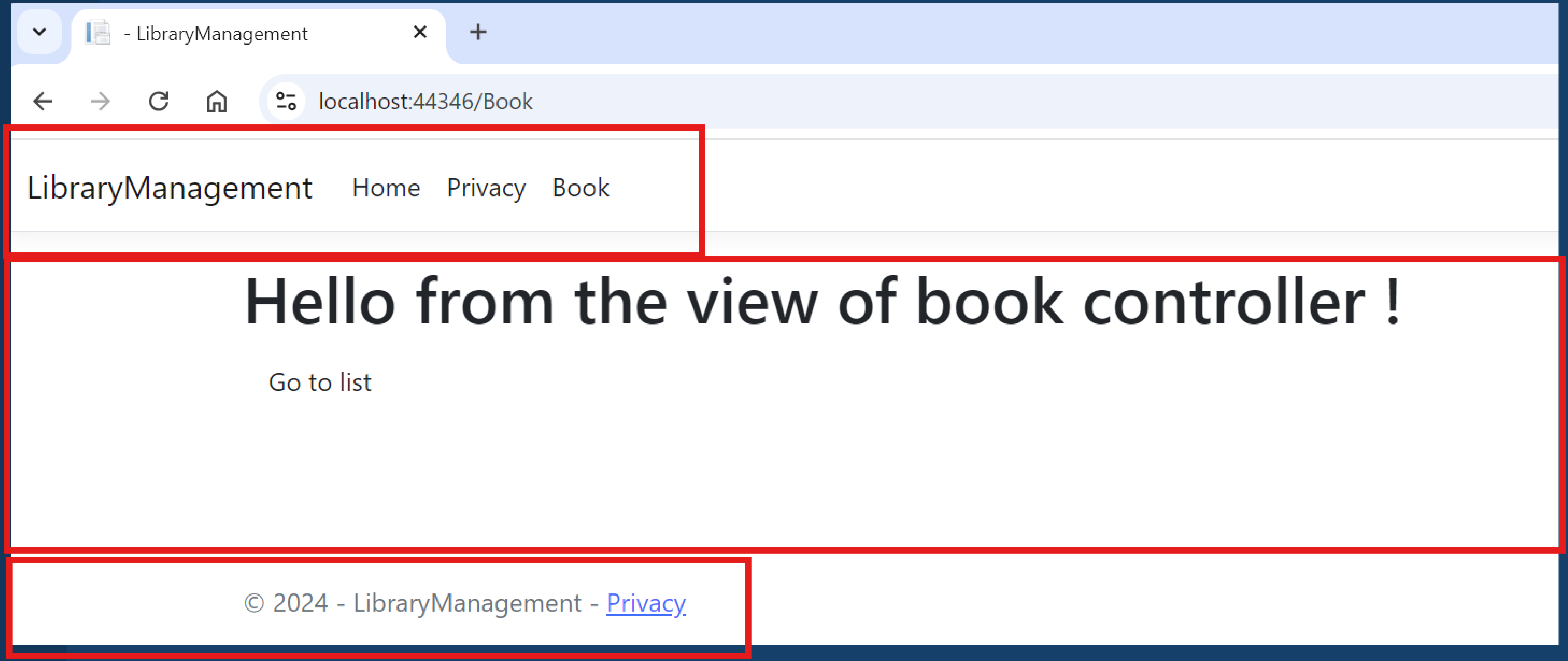
**Layout templates allow:**

**- Specifying the HTML container layout of a site in one place.**

**- Applying the HTML container layout across multiple pages in the site.**

ASP.NET Core MVC

## View – Change views and layout pages


- LibraryManagement

localhost:44346/Book

LibraryManagement Home Privacy Book

Hello from the view of book controller !

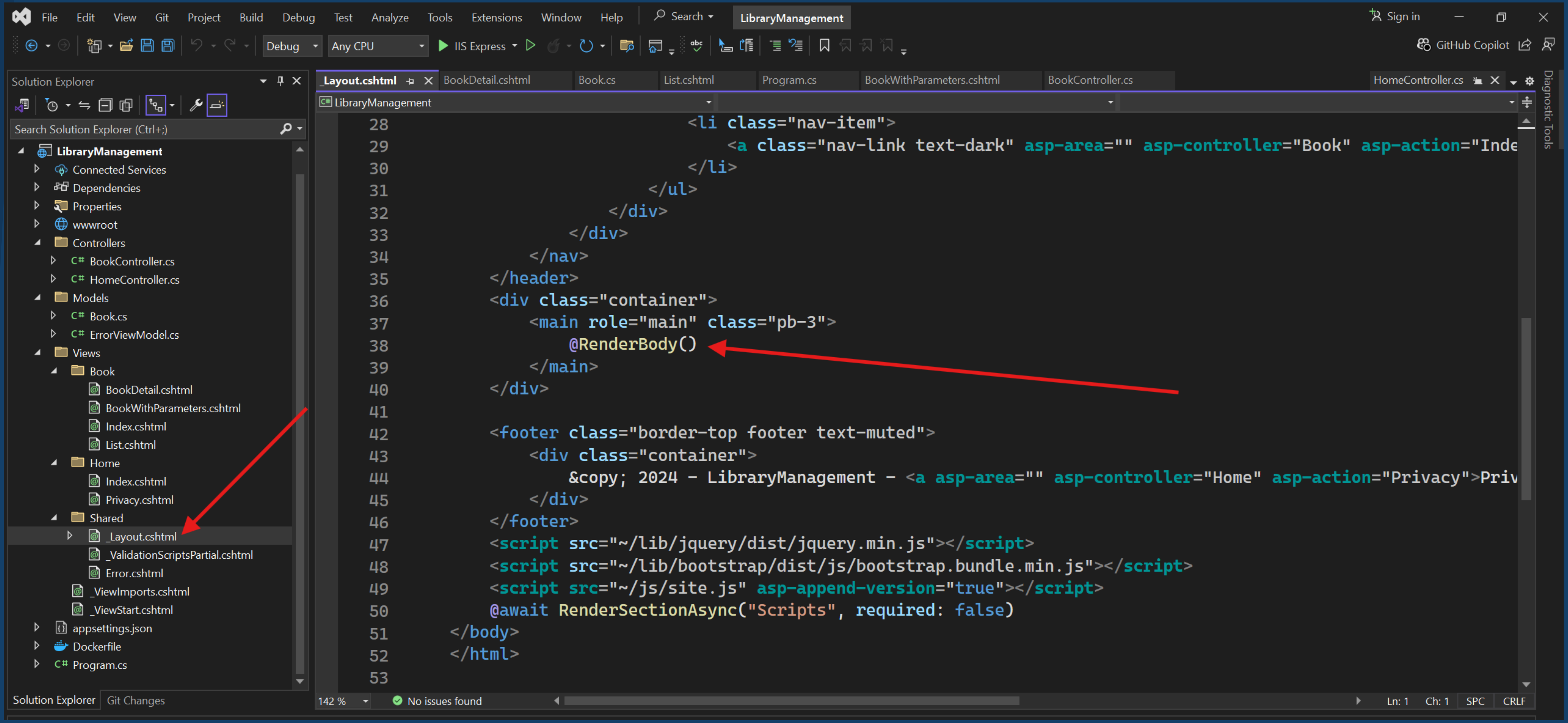
Go to list

© 2024 - LibraryManagement - Privacy

# ASP.NET Core MVC

## View – Change views and layout pages

RenderBody is a placeholder where all the view-specific pages you create show up, wrapped in the layout page. For example, if you select the Privacy link, the Views/Home/Privacy.cshtml view is rendered inside the RenderBody method.



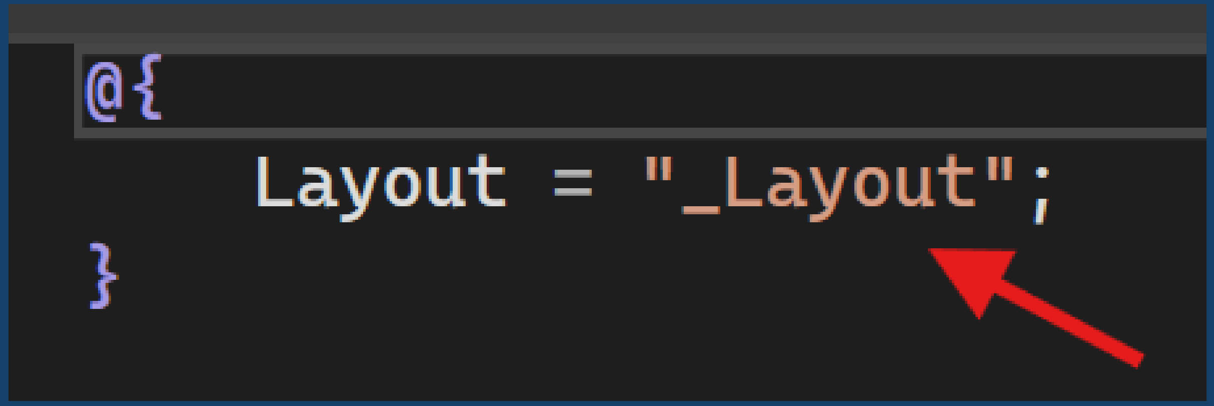
```
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-controller="Book" asp-action="Inde
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2024 - LibraryManagement - <a asp-area="" asp-controller="Home" asp-action="Privacy">Priv
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

# ASP.NET Core MVC

## View – Default layout

The Views/_ViewStart.cshtml file brings in the Views/Shared/_Layout.cshtml file to each view.

The Layout property can be used to set a different layout view, or set it to null so no layout file will be used.

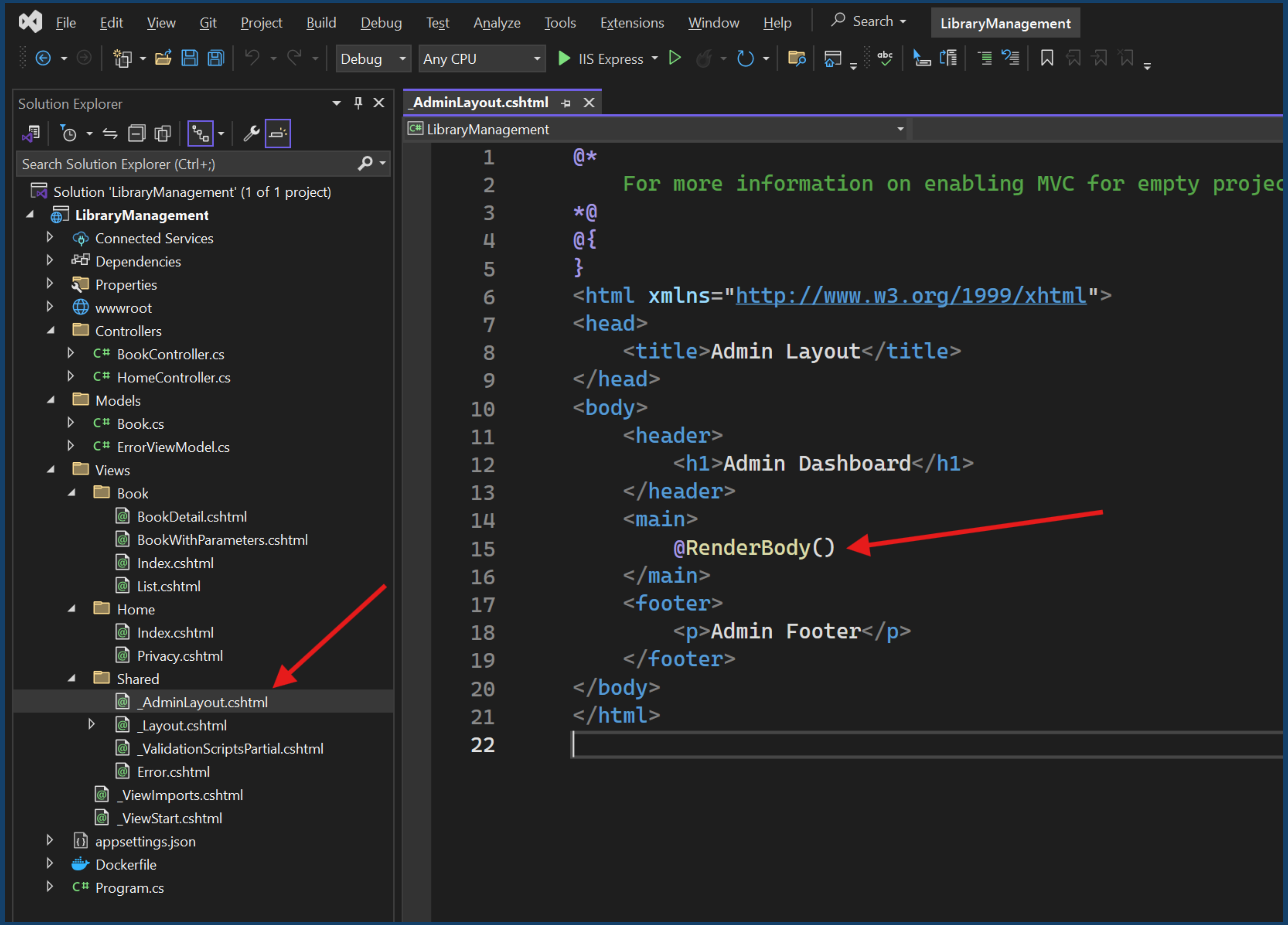


```
@{
    Layout = "_Layout";
}
```

# ASP.NET Core MVC

## View – Change the layout for each view

- First, create a new layout in the “shared” folder.



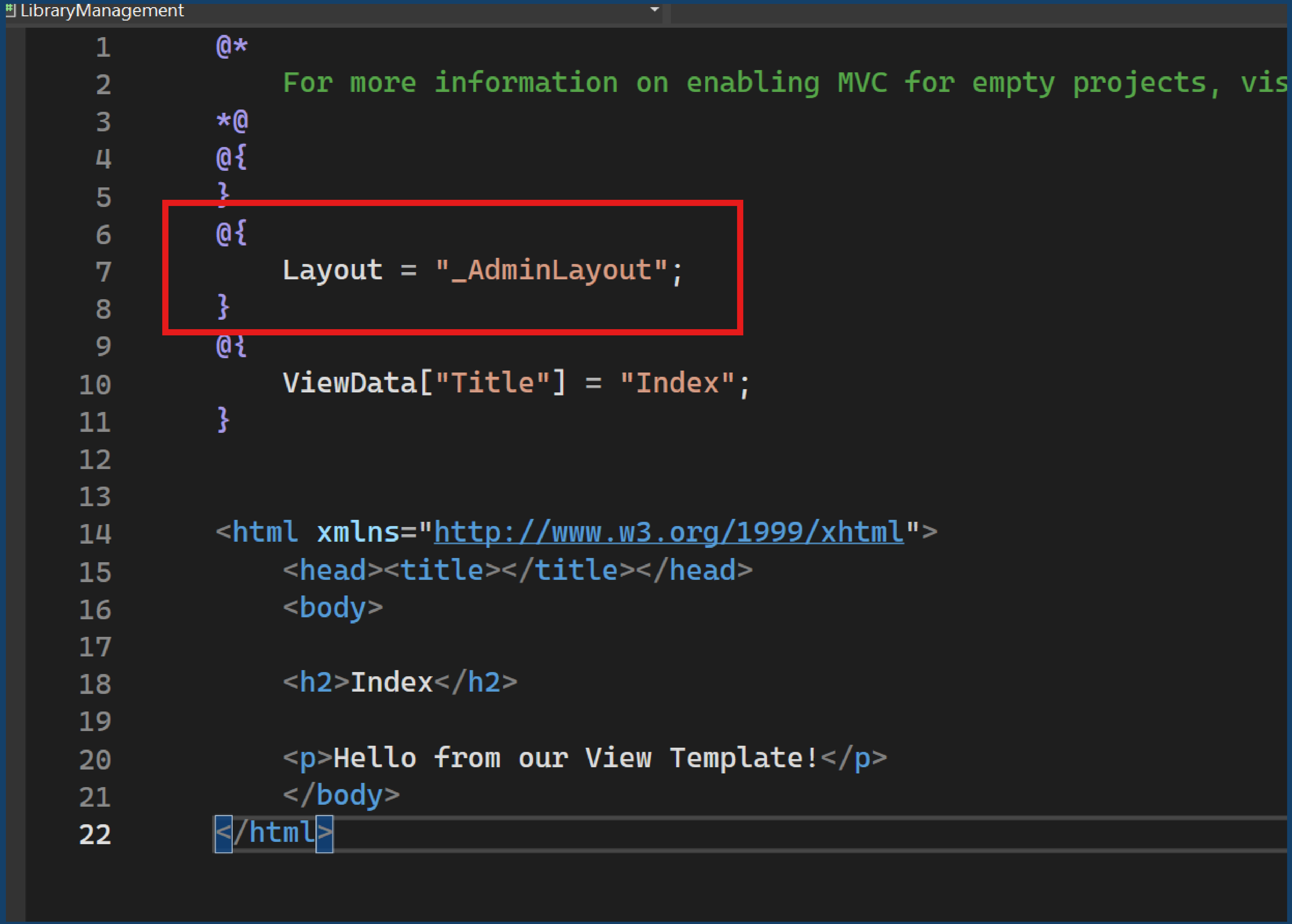
```
Solution 'LibraryManagement' (1 of 1 project)
  LibraryManagement
    Connected Services
    Dependencies
    Properties
    wwwroot
    Controllers
      BookController.cs
      HomeController.cs
    Models
      Book.cs
      ErrorViewModel.cs
    Views
      Book
        BookDetail.cshtml
        BookWithParameters.cshtml
        Index.cshtml
        List.cshtml
      Home
        Index.cshtml
        Privacy.cshtml
      Shared
        _AdminLayout.cshtml
        _Layout.cshtml
        _ValidationScriptsPartial.cshtml
        Error.cshtml
      _ViewImports.cshtml
      _ViewStart.cshtml
    appsettings.json
    Dockerfile
    Program.cs
```

```
@*
    For more information on enabling MVC for empty projec
*@
@{
}
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Admin Layout</title>
</head>
<body>
    <header>
        <h1>Admin Dashboard</h1>
    </header>
    <main>
        @RenderBody()
    </main>
    <footer>
        <p>Admin Footer</p>
    </footer>
</body>
</html>
```

# ASP.NET Core MVC

## View – Change the layout for each view

- Add a layout for each view. If a view doesn't have a layout set, the default layout will be applied.



LibraryManagement

```
@*
    For more information on enabling MVC for empty projects, vis
*@
@{
}
@{
    Layout = "_AdminLayout";
}
@{
    ViewData["Title"] = "Index";
}


<html xmlns="http://www.w3.org/1999/xhtml">
    <head><title></title></head>
    <body>

    <h2>Index</h2>

    <p>Hello from our View Template!</p>
    </body>
</html>
```

# ASP.NET Core MVC

## View – Passing Data from the Controller to the View

- Controller actions are invoked in response to an incoming URL request. A controller class is where the code is written that handles the incoming browser requests. The controller retrieves data from a data source and decides what type of response to send back to the browser. View templates can be used from a controller to generate and format an HTML response to the browser.

- Controllers are responsible for providing the data required in order for a view template to render a response.

- View templates should not:
- Do business logic

Interact with a database directly.

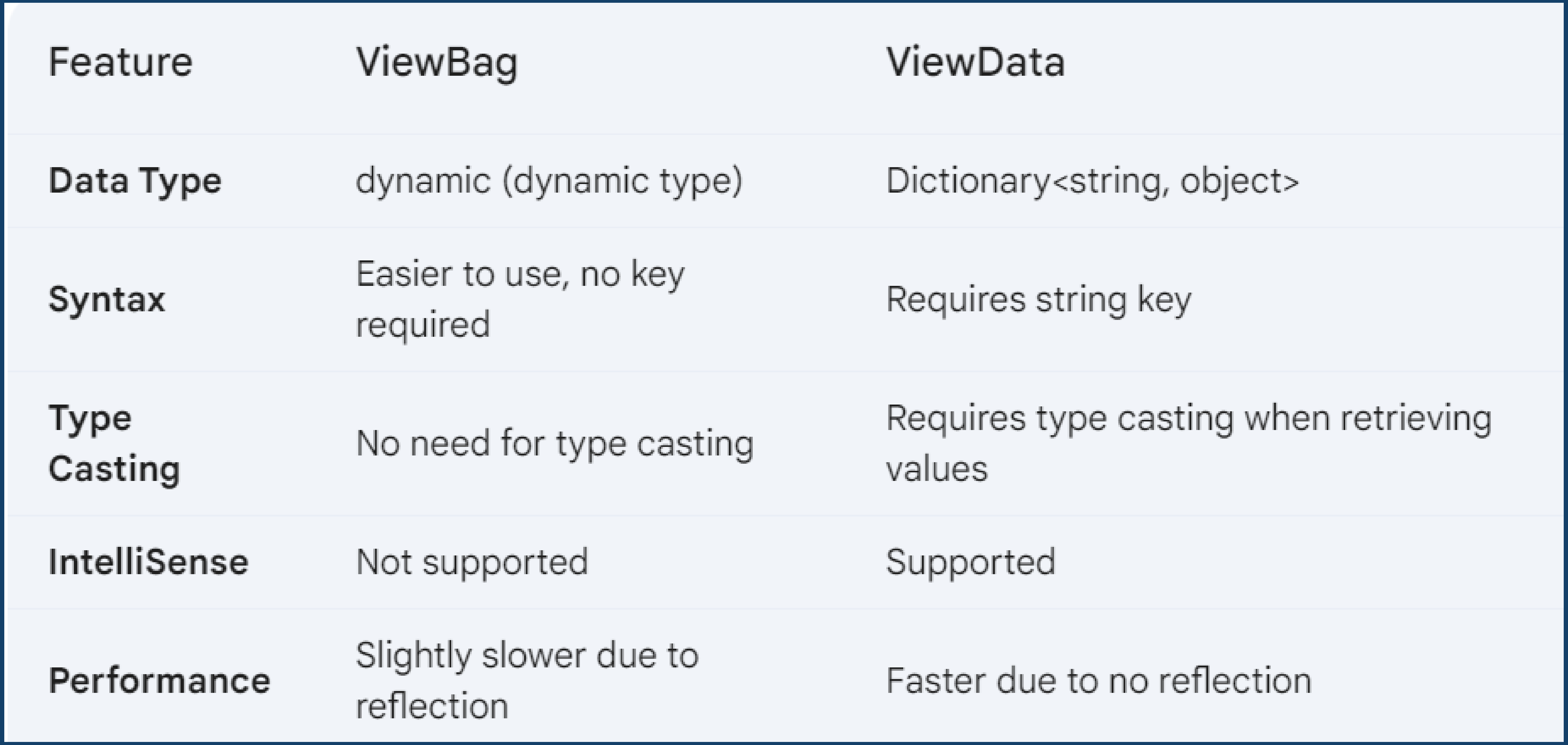
- A view template should work only with the data that's provided to it by the controller. Maintaining this "separation of concerns" helps keep the code:

- Clean.
- Testable.
- Maintainable.

# ASP.NET Core MVC

## View – Passing Data from the Controller to the View

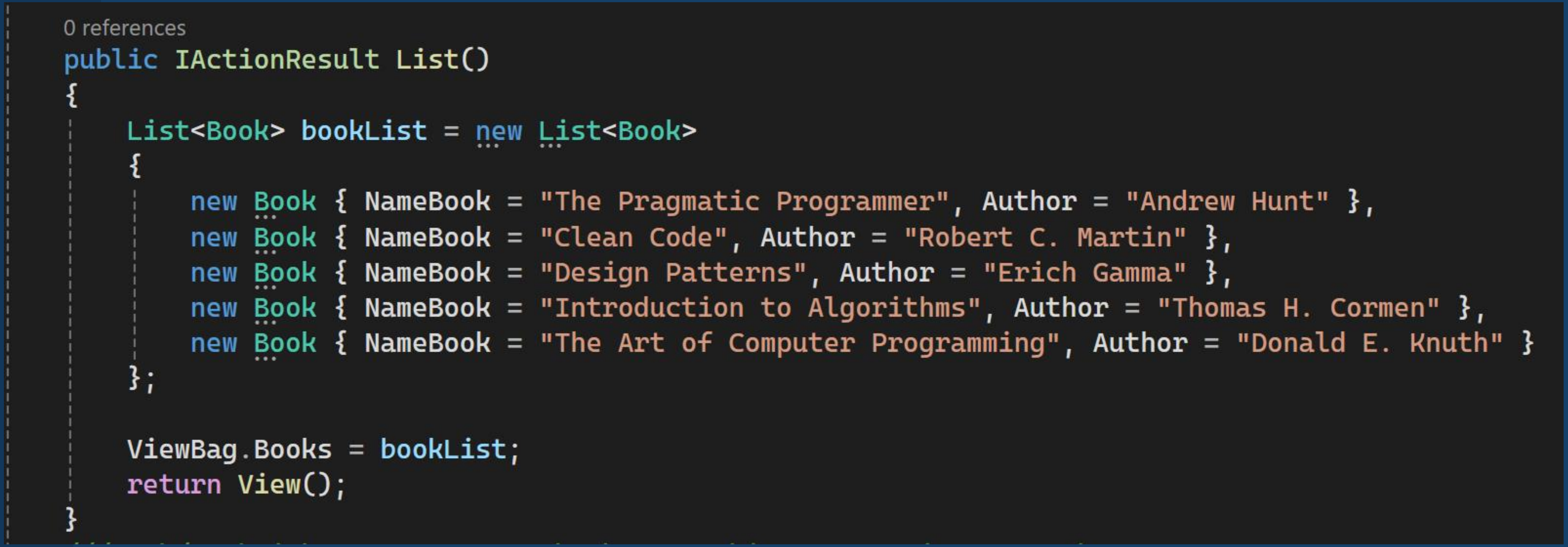
- ViewData and ViewBag have a same way to use that. Both of it use to pass the data from controller to the view.



| Feature | ViewBag | ViewData |
|---|---|---|
| **Data Type** | dynamic (dynamic type) | Dictionary<string, object> |
| **Syntax** | Easier to use, no key required | Requires string key |
| **Type Casting** | No need for type casting | Requires type casting when retrieving values |
| **IntelliSense** | Not supported | Supported |
| **Performance** | Slightly slower due to reflection | Faster due to no reflection |

## View – Passing Data from the Controller to the View

- Prepare data in controller

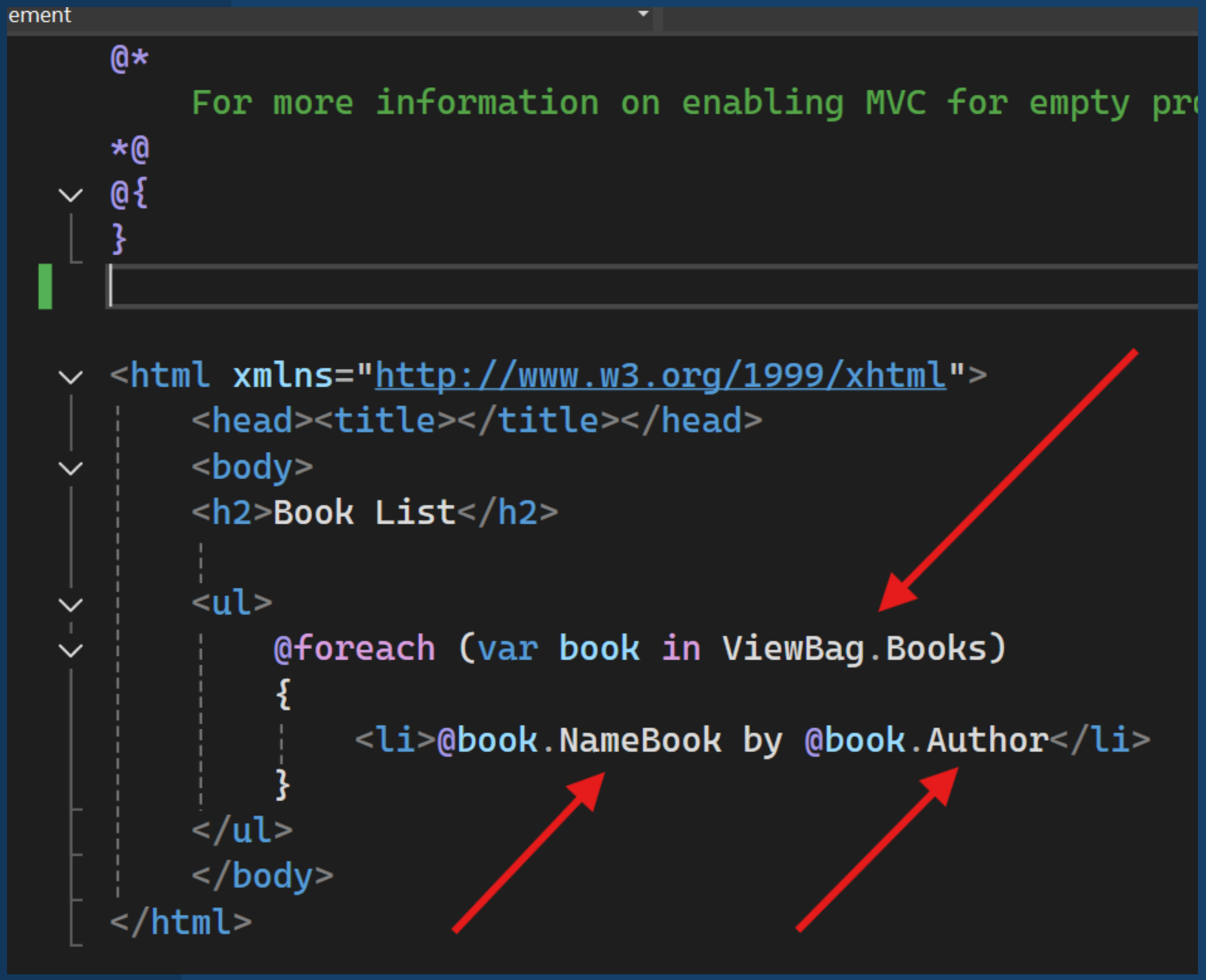


```
0 references
public IActionResult List()
{
    List<Book> bookList = new List<Book>
    {
        new Book { NameBook = "The Pragmatic Programmer", Author = "Andrew Hunt" },
        new Book { NameBook = "Clean Code", Author = "Robert C. Martin" },
        new Book { NameBook = "Design Patterns", Author = "Erich Gamma" },
        new Book { NameBook = "Introduction to Algorithms", Author = "Thomas H. Cormen" },
        new Book { NameBook = "The Art of Computer Programming", Author = "Donald E. Knuth" }
    };

    ViewBag.Books = bookList;
    return View();
}
```

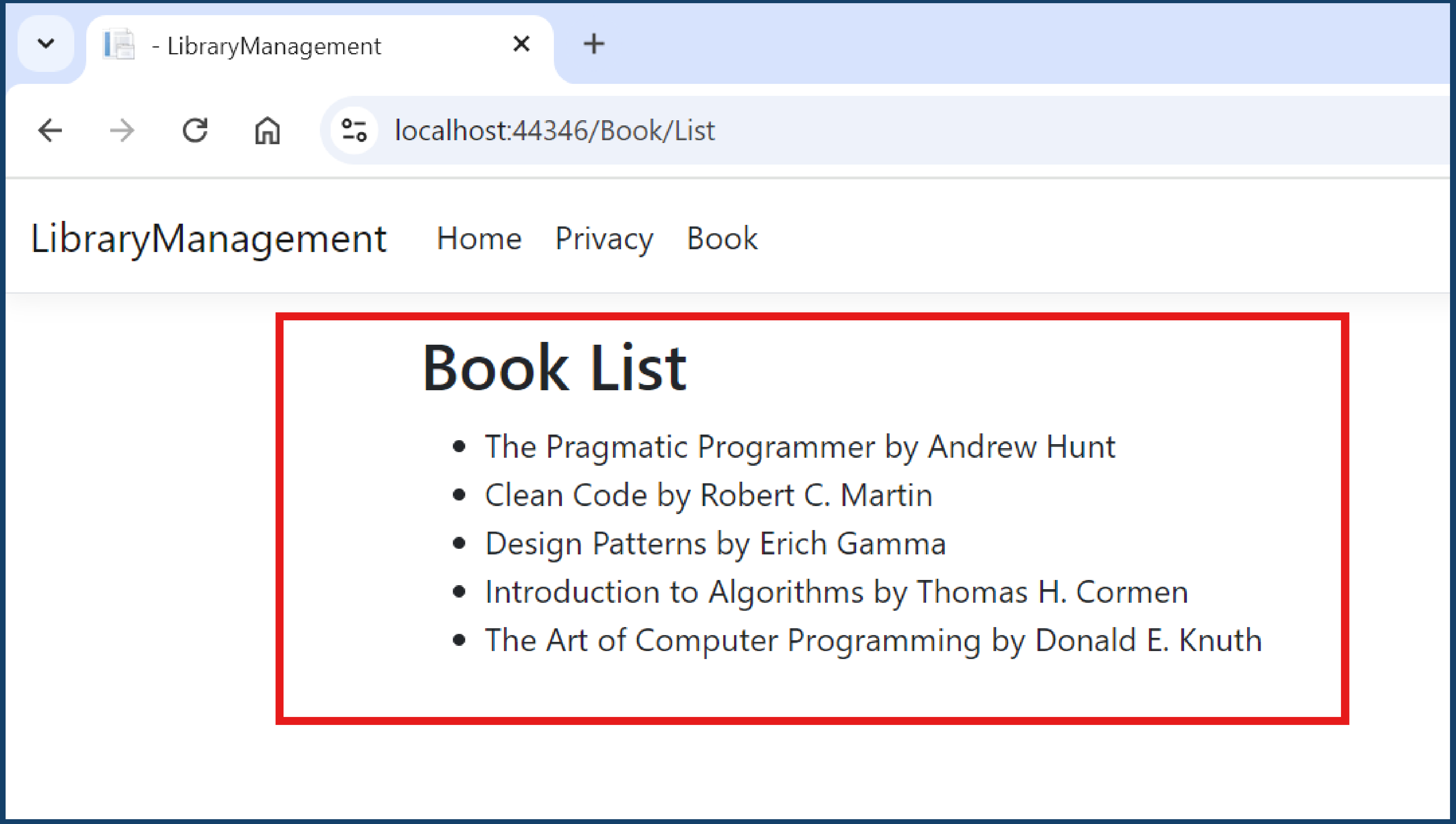
# ASP.NET Core MVC

## View – Passing Data from the Controller to the View

- Show the data get from controller to the view



```
@*
    For more information on enabling MVC for empty pr
*@
@{
}

<html xmlns="http://www.w3.org/1999/xhtml">
    <head><title></title></head>
    <body>
    <h2>Book List</h2>

    <ul>
        @foreach (var book in ViewBag.Books)
        {
            <li>@book.NameBook by @book.Author</li>
        }
    </ul>
    </body>
</html>
```



Book List

- The Pragmatic Programmer by Andrew Hunt
- Clean Code by Robert C. Martin
- Design Patterns by Erich Gamma
- Introduction to Algorithms by Thomas H. Cormen
- The Art of Computer Programming by Donald E. Knuth

## ASP.NET Core MVC

# Model

## ASP.NET Core MVC

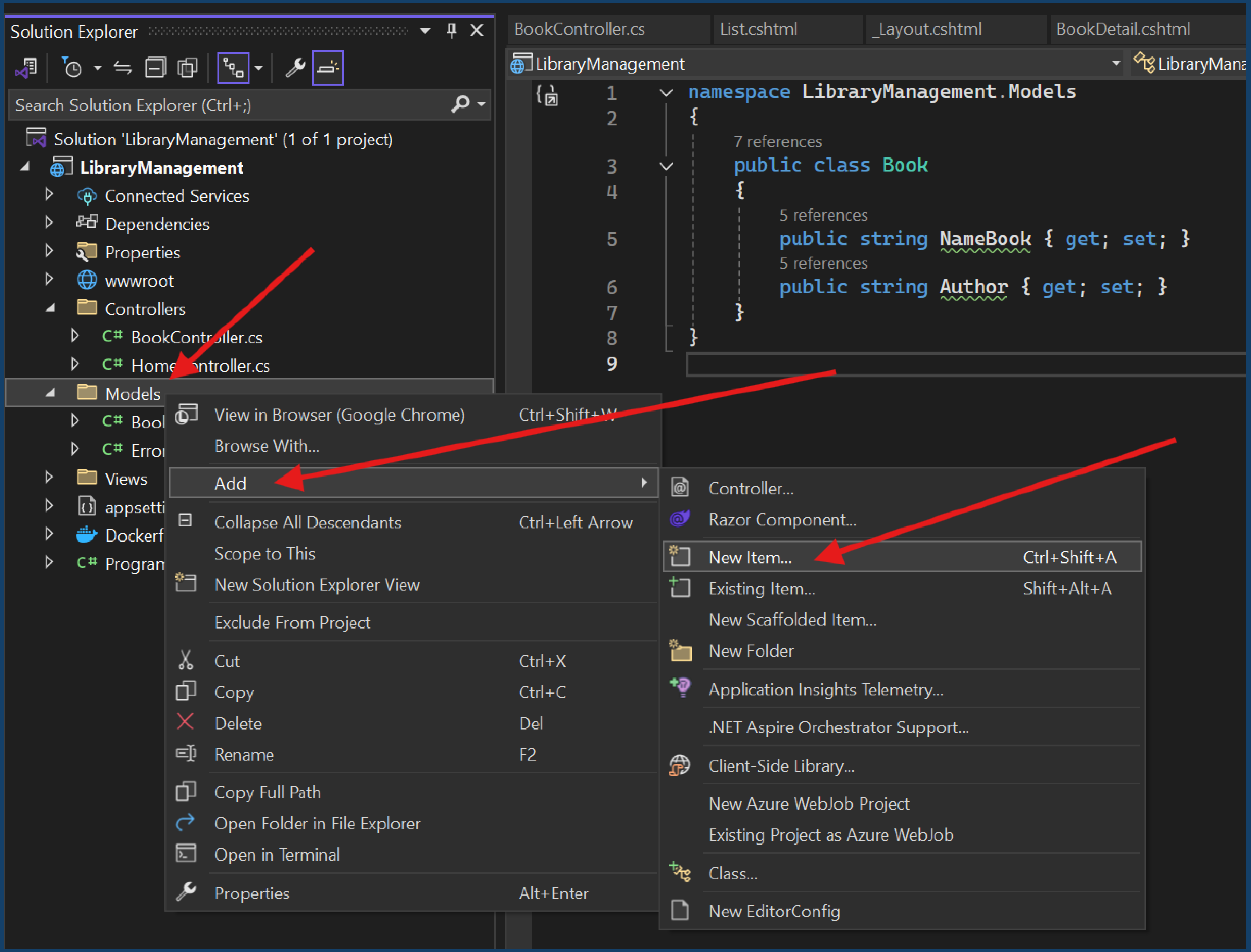
### Model

These model classes are used with Entity Framework Core (EF Core) to work with a database. EF Core is an object-relational mapping (ORM) framework that simplifies the data access code that you have to write.

The model classes created are known as POCO classes, from Plain Old CLR Objects. POCO classes don't have any dependency on EF Core. They only define the properties of the data to be stored in the database.

# ASP.NET Core MVC

## Model – Add a model to an ASP.NET Core MVC app



```
namespace LibraryManagement.Models
{
    7 references
    public class Book
    {
        5 references
        public string NameBook { get; set; }
        5 references
        public string Author { get; set; }
    }
}
```

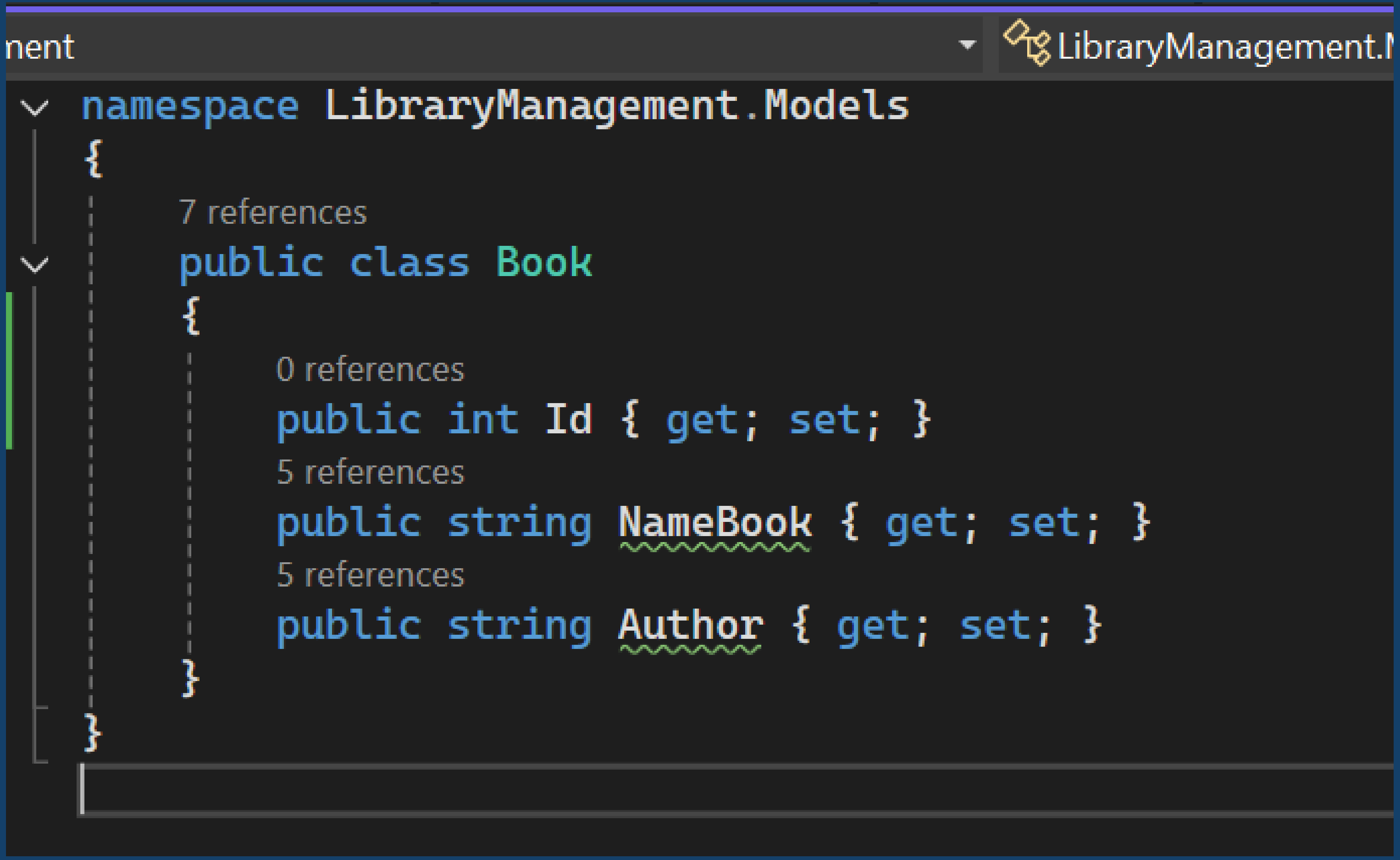
# ASP.NET Core MVC

## Model – Add a model to an ASP.NET Core MVC app

The **Book** class contains an Id field, which is required by the database for the primary key.

The DataType attribute on ReleaseDate specifies the type of the data (Date). With this attribute:

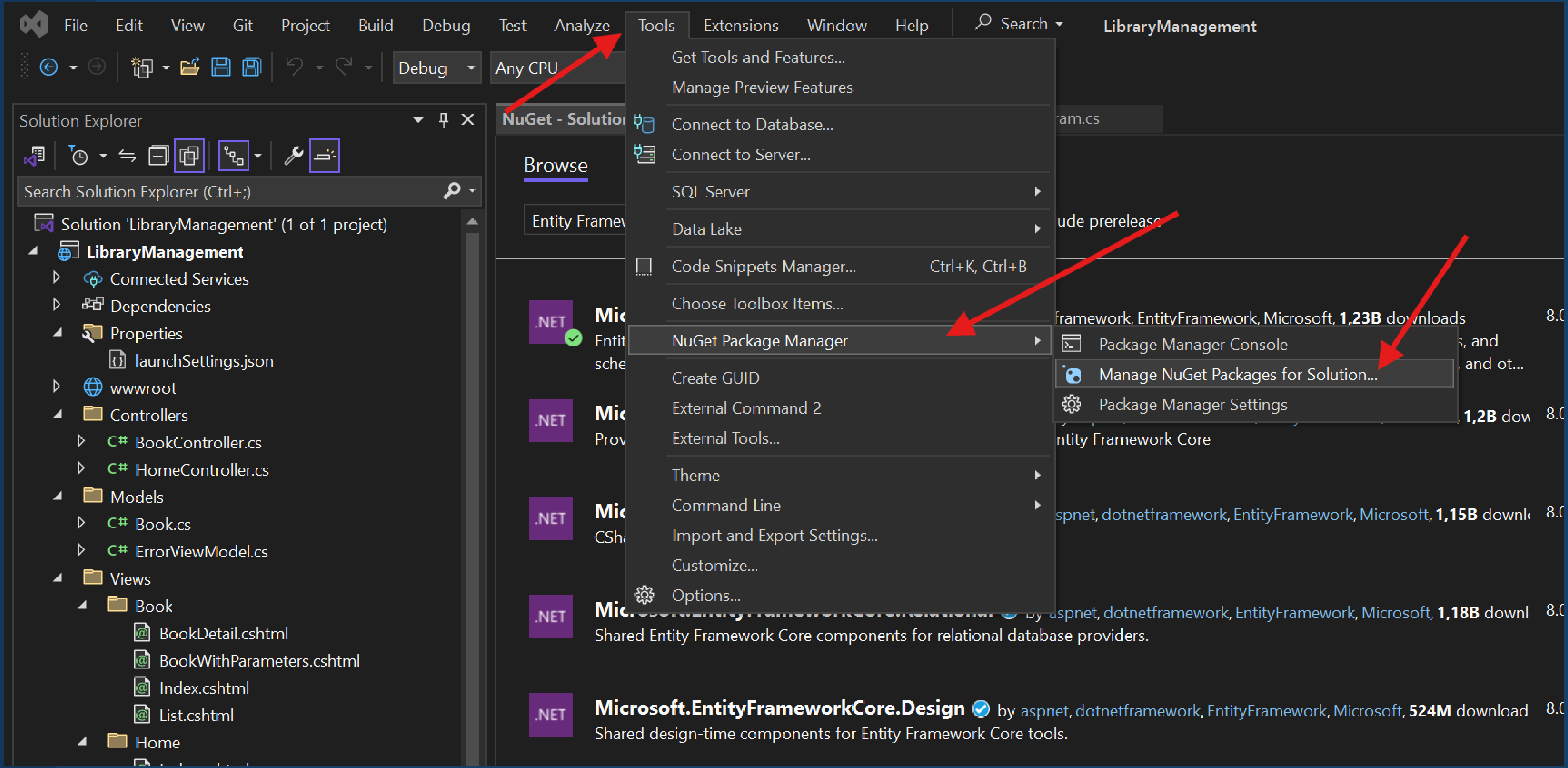
- The user isn't required to enter time information in the date field.
- Only the date is displayed, not time information.



```
namespace LibraryManagement.Models
{
    7 references
    public class Book
    {
        0 references
        public int Id { get; set; }
        5 references
        public string NameBook { get; set; }
        5 references
        public string Author { get; set; }
    }
}
```

# ASP.NET Core MVC

## Model – Entity Framework

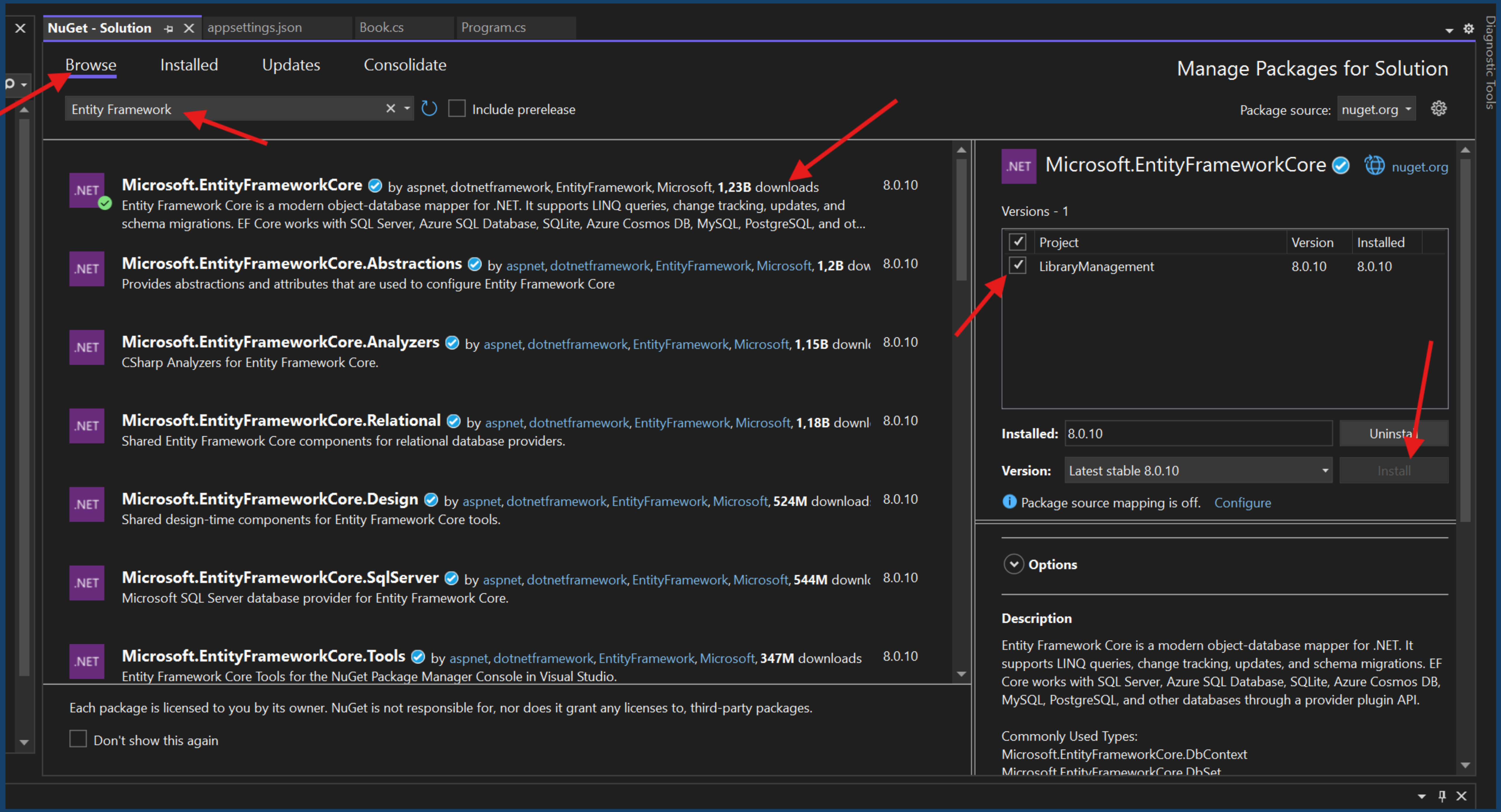


# ASP.NET Core MVC

## Model – Entity Framework

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore



Microsoft.EntityFrameworkCore by aspnet, dotnetframework, EntityFramework, Microsoft, 1,23B downloads 8.0.10
Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and ot...

Microsoft.EntityFrameworkCore.Abstractions by aspnet, dotnetframework, EntityFramework, Microsoft, 1,2B dow 8.0.10
Provides abstractions and attributes that are used to configure Entity Framework Core

Microsoft.EntityFrameworkCore.Analyzers by aspnet, dotnetframework, EntityFramework, Microsoft, 1,15B downl 8.0.10
CSharp Analyzers for Entity Framework Core.

Microsoft.EntityFrameworkCore.Relational by aspnet, dotnetframework, EntityFramework, Microsoft, 1,18B downl 8.0.10
Shared Entity Framework Core components for relational database providers.

Microsoft.EntityFrameworkCore.Design by aspnet, dotnetframework, EntityFramework, Microsoft, 524M download 8.0.10
Shared design-time components for Entity Framework Core tools.

Microsoft.EntityFrameworkCore.SqlServer by aspnet, dotnetframework, EntityFramework, Microsoft, 544M downl 8.0.10
Microsoft SQL Server database provider for Entity Framework Core.

Microsoft.EntityFrameworkCore.Tools by aspnet, dotnetframework, EntityFramework, Microsoft, 347M downloads 8.0.10
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

☐ Don't show this again

Microsoft.EntityFrameworkCore nuget.org

Versions - 1

| ✓ Project | Version | Installed |
|---|---|---|
| ✓ LibraryManagement | 8.0.10 | 8.0.10 |

Installed: 8.0.10 Uninstall

Version: Latest stable 8.0.10 Install

Package source mapping is off. Configure

Options

**Description**

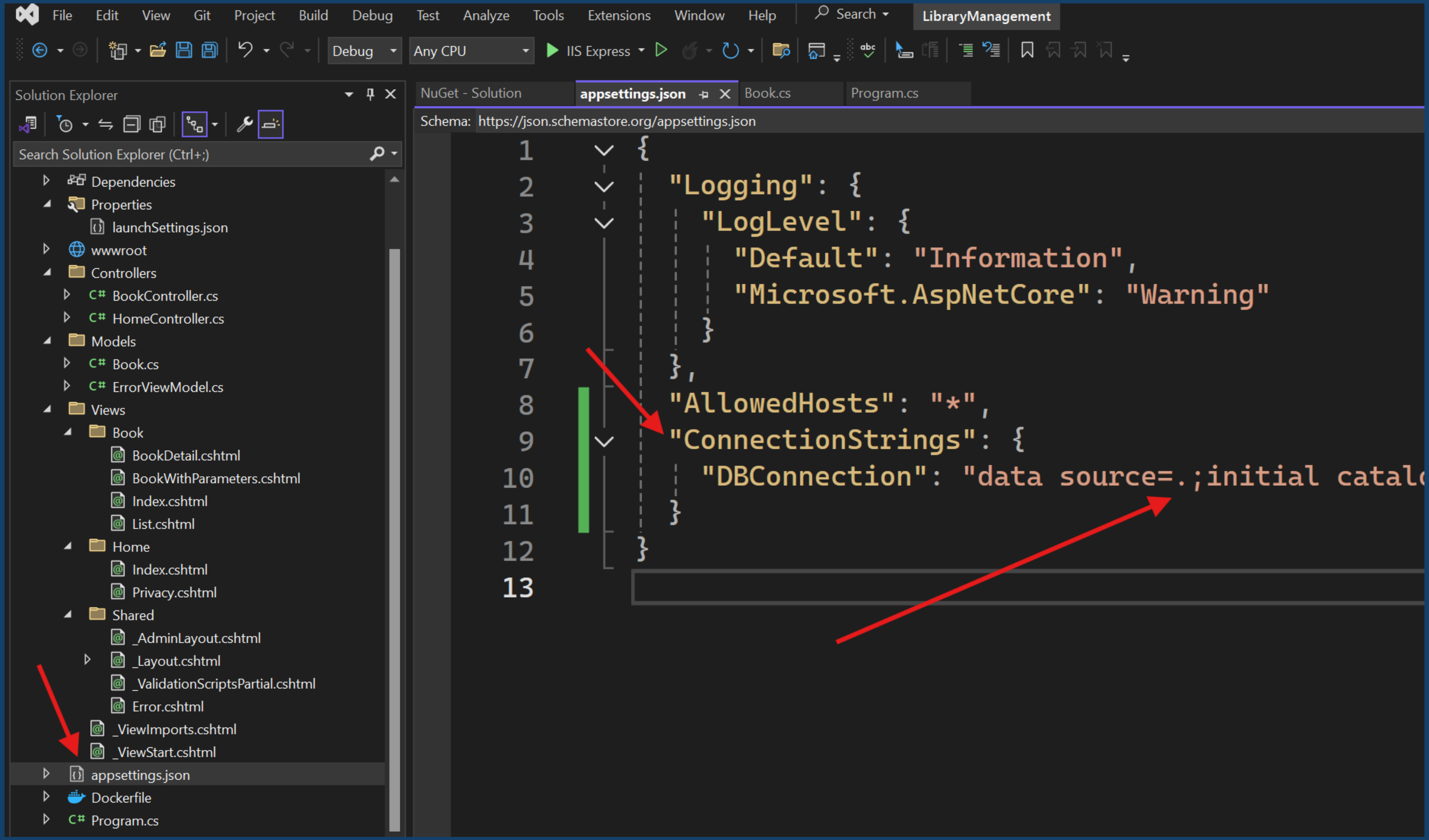
Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other databases through a provider plugin API.

Commonly Used Types:
Microsoft.EntityFrameworkCore.DbContext
Microsoft.EntityFrameworkCore.DbSet

# ASP.NET Core MVC

## Model – appsetting.json

data source=.;initial catalog=YourDatabaseName;persist security info=True;user id=sa;password=123456;MultipleActiveResultSets=True;encrypt=false



Schema: https://json.schemastore.org/appsettings.json

```json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DBConnection": "data source=.;initial catal
  }
}
```

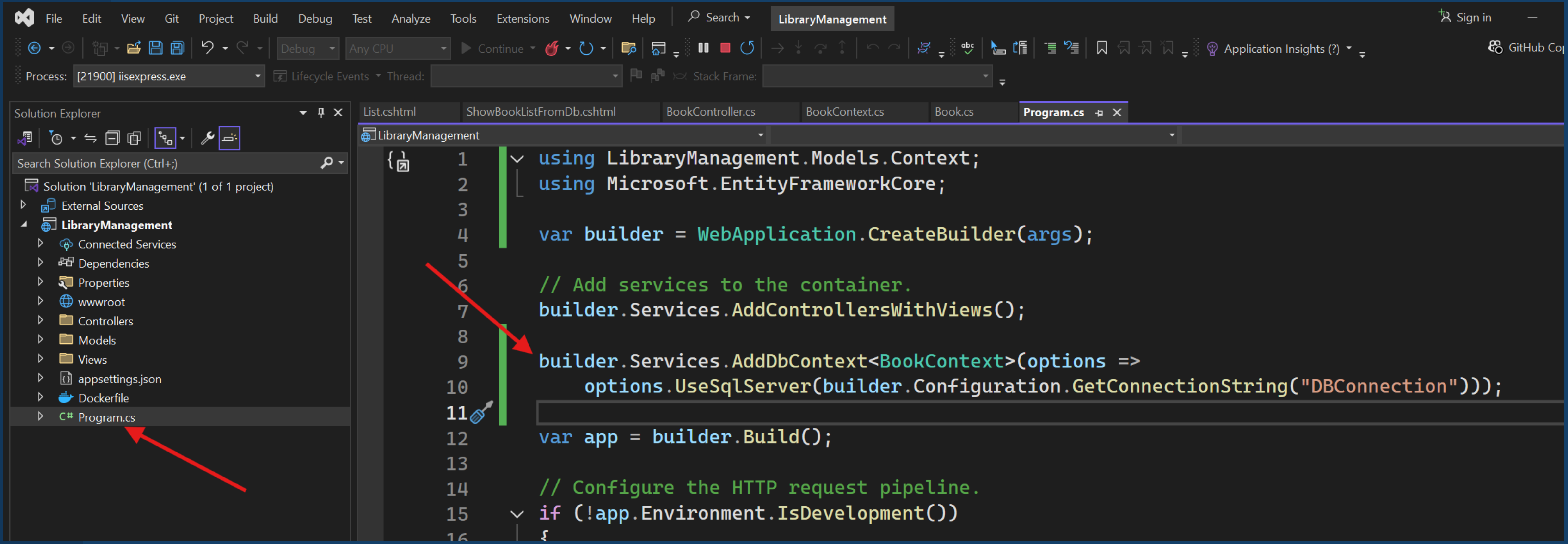
## ASP.NET Core MVC

**Model – Dependency injection**

- ASP.NET Core is built with dependency injection (DI). Services, such as the database context, are registered with DI in Program.cs. These services are provided to components that require them via constructor parameters.
- In the Controllers/MoviesController.cs file, the constructor uses Dependency Injection to inject the MvcMovieContext database context into the controller. The database context is used in each of the CRUD methods in the controller.
- Scaffolding generated the following highlighted code in Program.cs:

# ASP.NET Core MVC

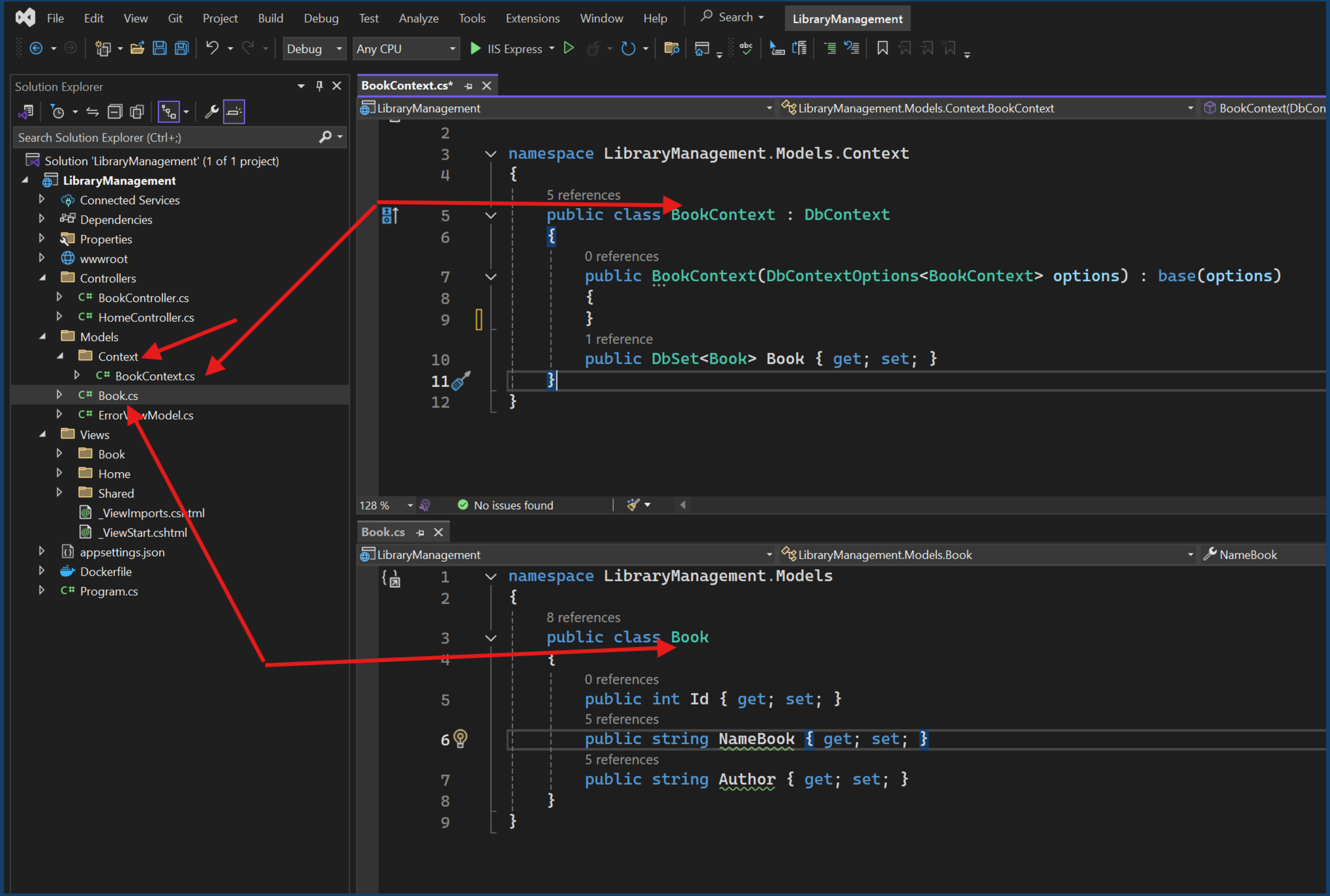
## Model – Dependency injection



```
using LibraryManagement.Models.Context;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

builder.Services.AddDbContext<BookContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DBConnection")));

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
```

# ASP.NET Core MVC

## Model – Dbcontext



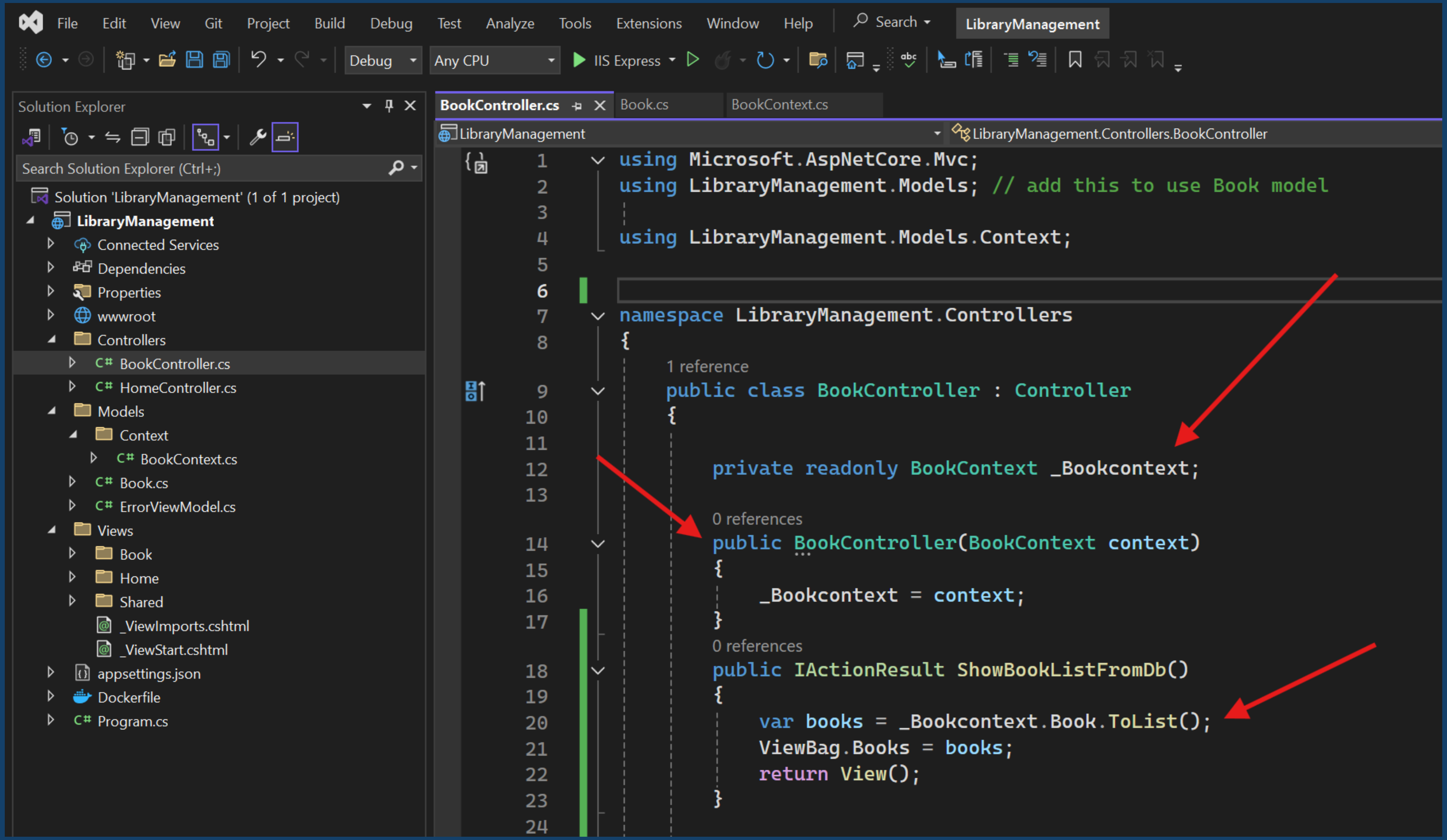
```
namespace LibraryManagement.Models.Context
{
    public class BookContext : DbContext
    {
        public BookContext(DbContextOptions<BookContext> options) : base(options)
        {
        }
        public DbSet<Book> Book { get; set; }
    }
}
```

```
namespace LibraryManagement.Models
{
    public class Book
    {
        public int Id { get; set; }
        public string NameBook { get; set; }
        public string Author { get; set; }
    }
}
```

EIU

# ASP.NET Core MVC

## Model – Dependency injection in the controller

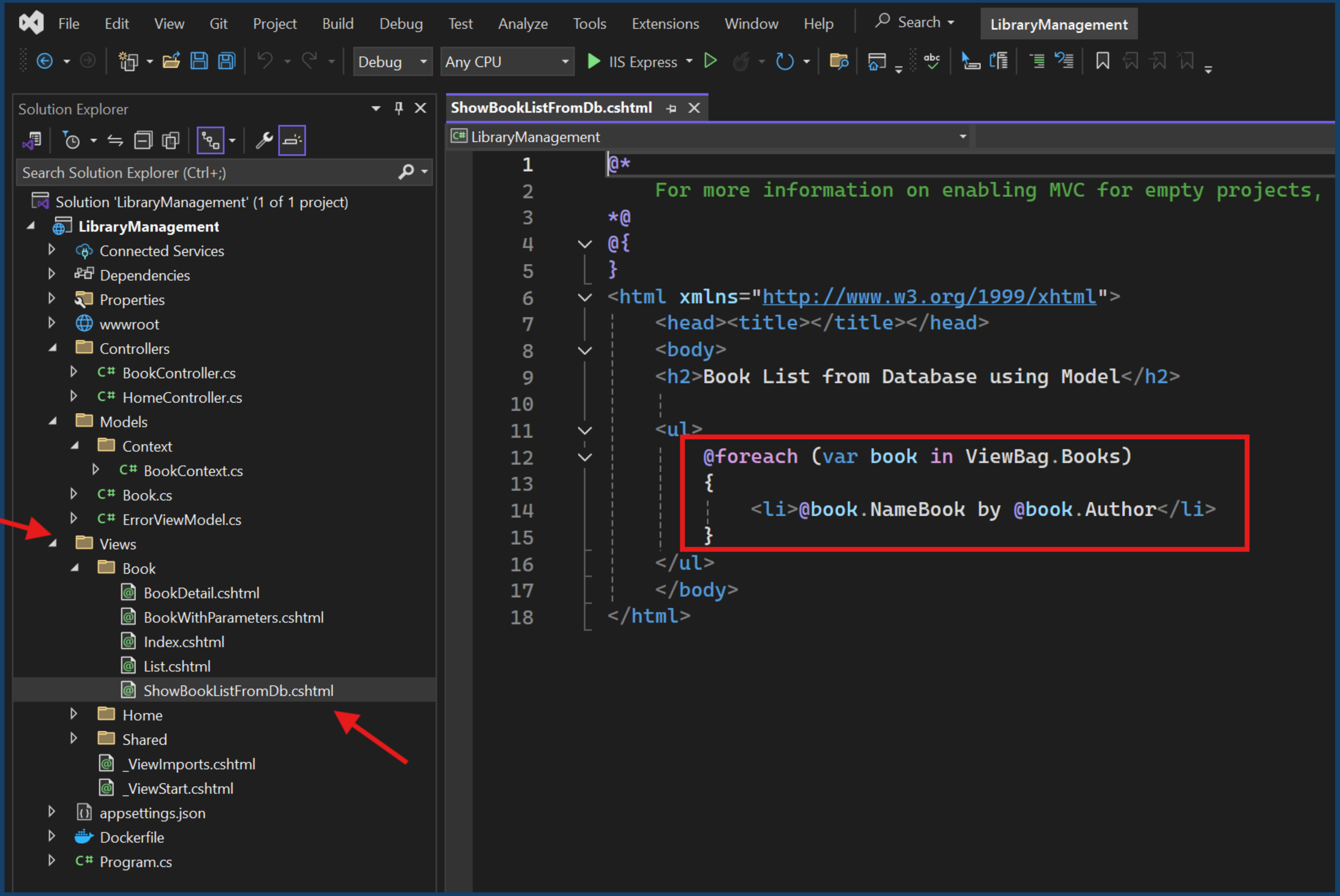


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search LibraryManagement
Debug Any CPU IIS Express
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'LibraryManagement' (1 of 1 project)
LibraryManagement
Connected Services
Dependencies
Properties
wwwroot
Controllers
BookController.cs
HomeController.cs
Models
Context
BookContext.cs
Book.cs
ErrorViewModel.cs
Views
Book
Home
Shared
_ViewImports.cshtml
_ViewStart.cshtml
appsettings.json
Dockerfile
Program.cs
BookController.cs Book.cs BookContext.cs
LibraryManagement
LibraryManagement.Controllers.BookController

```
using Microsoft.AspNetCore.Mvc;
using LibraryManagement.Models; // add this to use Book model

using LibraryManagement.Models.Context;


namespace LibraryManagement.Controllers
{
    1 reference
    public class BookController : Controller
    {

        private readonly BookContext _Bookcontext;

        0 references
        public BookController(BookContext context)
        {
            _Bookcontext = context;
        }
        0 references
        public IActionResult ShowBookListFromDb()
        {
            var books = _Bookcontext.Book.ToList();
            ViewBag.Books = books;
            return View();
        }
```

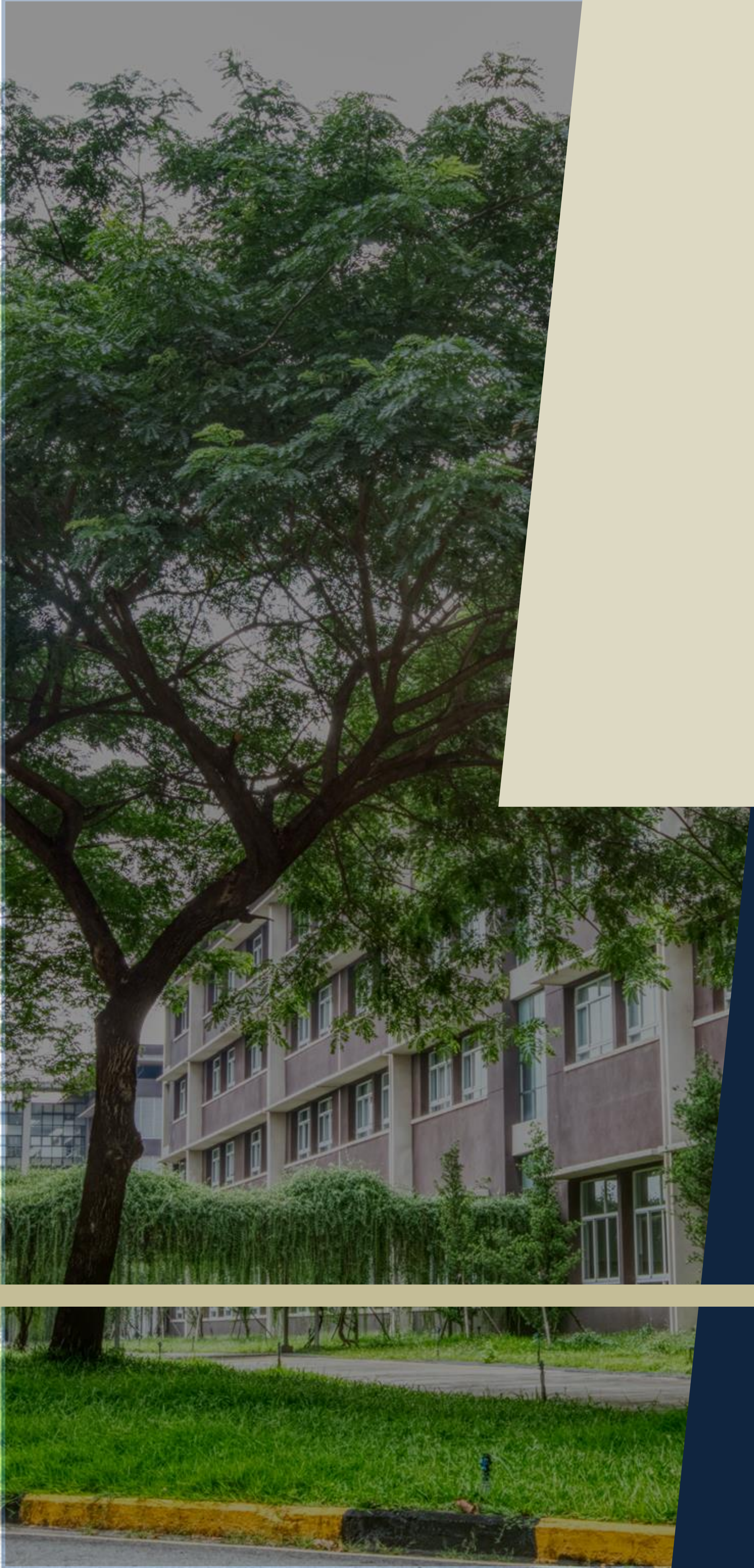
# ASP.NET Core MVC

## Model – Show in view



```
@*
    For more information on enabling MVC for empty projects,
*@
@{
}
<html xmlns="http://www.w3.org/1999/xhtml">
    <head><title></title></head>
    <body>
    <h2>Book List from Database using Model</h2>

    <ul>
        @foreach (var book in ViewBag.Books)
        {
            <li>@book.NameBook by @book.Author</li>
        }
    </ul>
    </body>
</html>
```

EIU
TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG
EASTERN
INTERNATIONAL
UNIVERSITY



*Start your future at EIU*

# Thank You