EIU
TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG
EASTERN
INTERNATIONAL
UNIVERSITY

.NET Programming

Chapter 2: ASP.NET Core MVC

EIU

TABLE CONTENT

1. Introduction to .Net Framework, .Net Core, ASP.Net, and ASP.Net Core
2. ASP.Net Core Get Started
3. MVC Model

.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

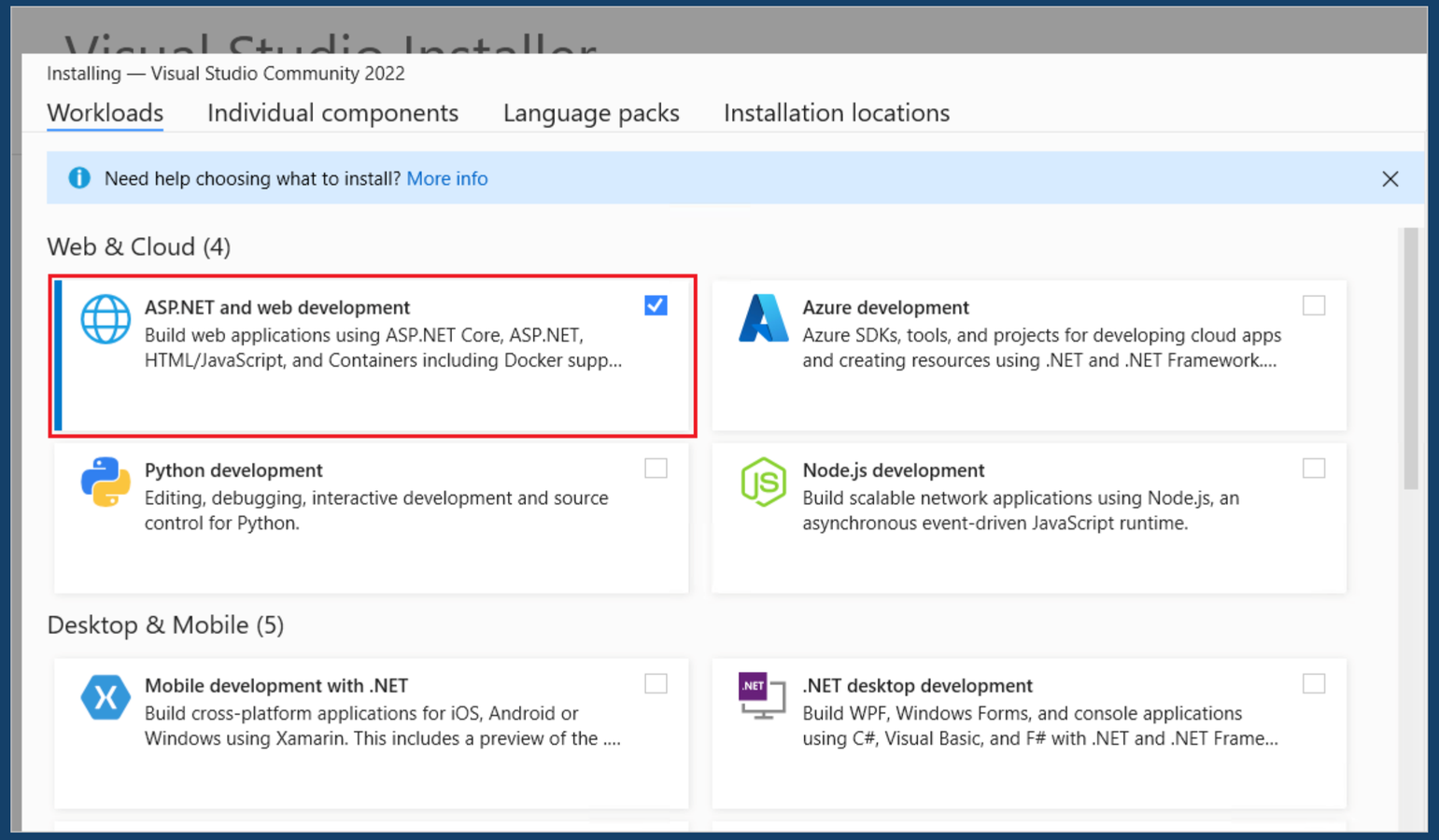
	ASP.NET	ASP.NET Core
Supported Platforms	Windows only	Cross-platform: Windows, macOS, Linux
Source Code	Primarily closed-source	Open-source, community-driven
Performance	Lower performance compared to ASP.NET Core	High performance due to optimization and lightweight design
Architecture	Monolithic: Integrates 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

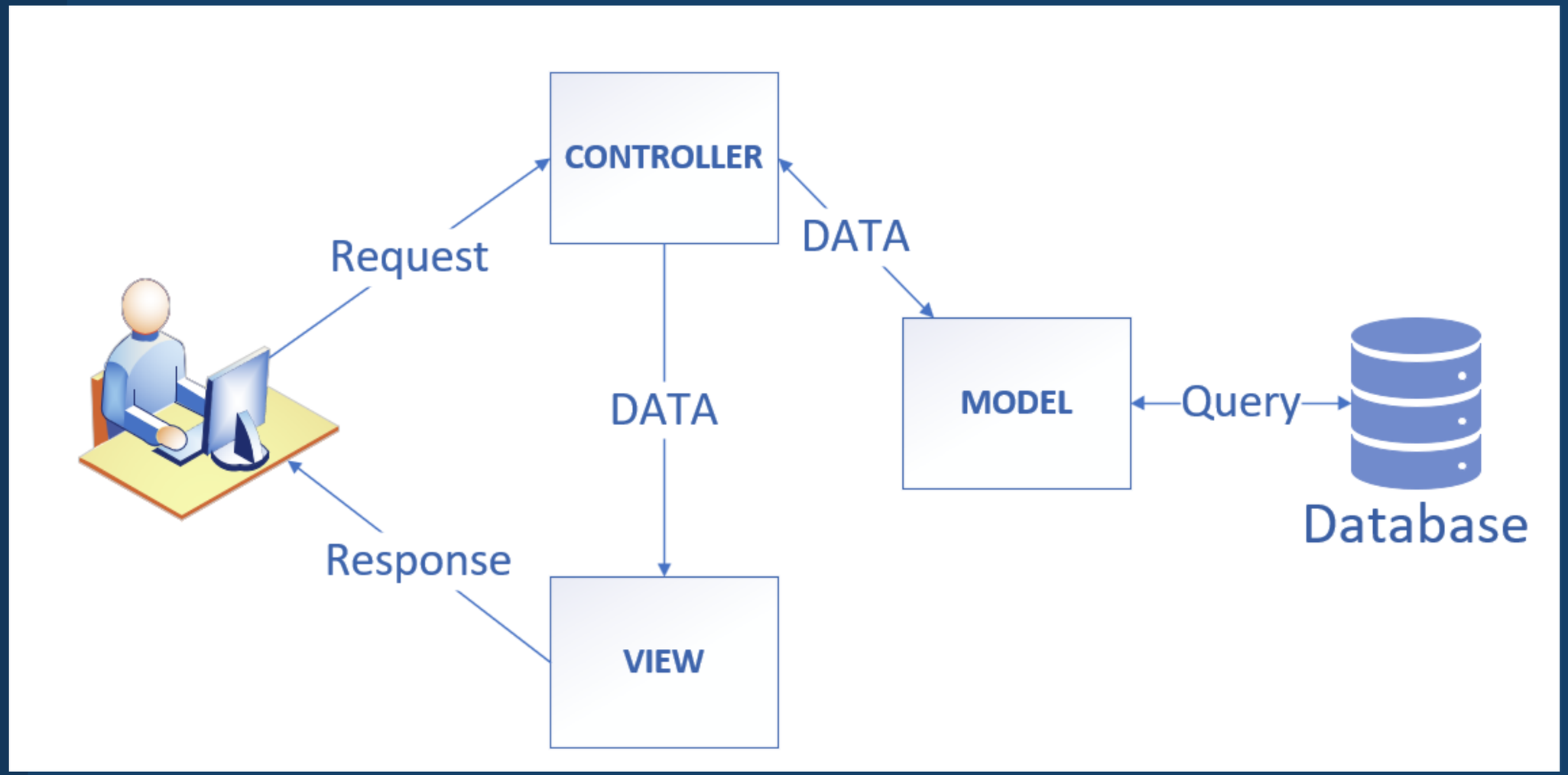
ASP.NET Core MVC

Prerequisites: .NET Core 8.0



ASP.NET Core MVC

MVC Architecture:



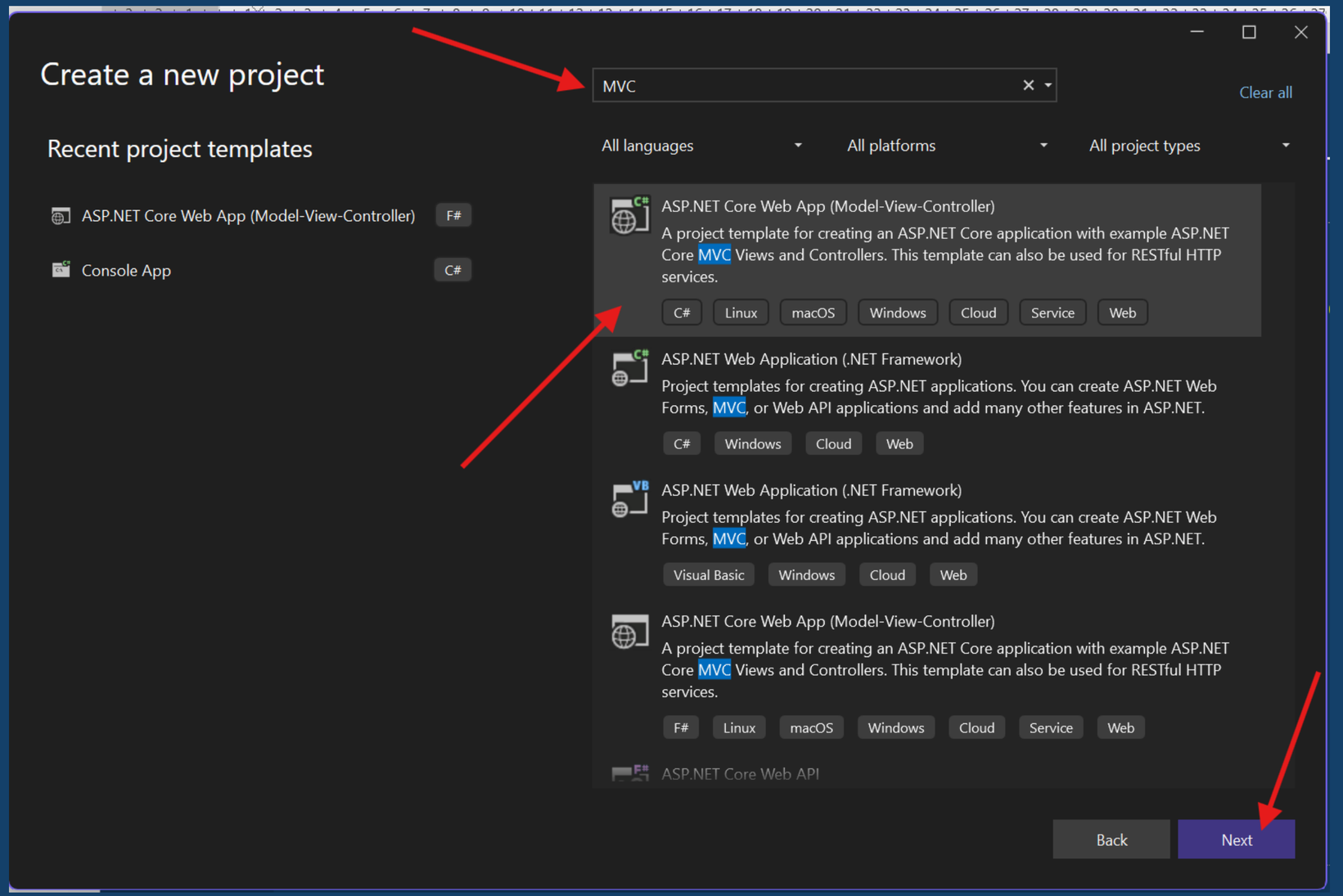
ASP.NET Core MVC

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.

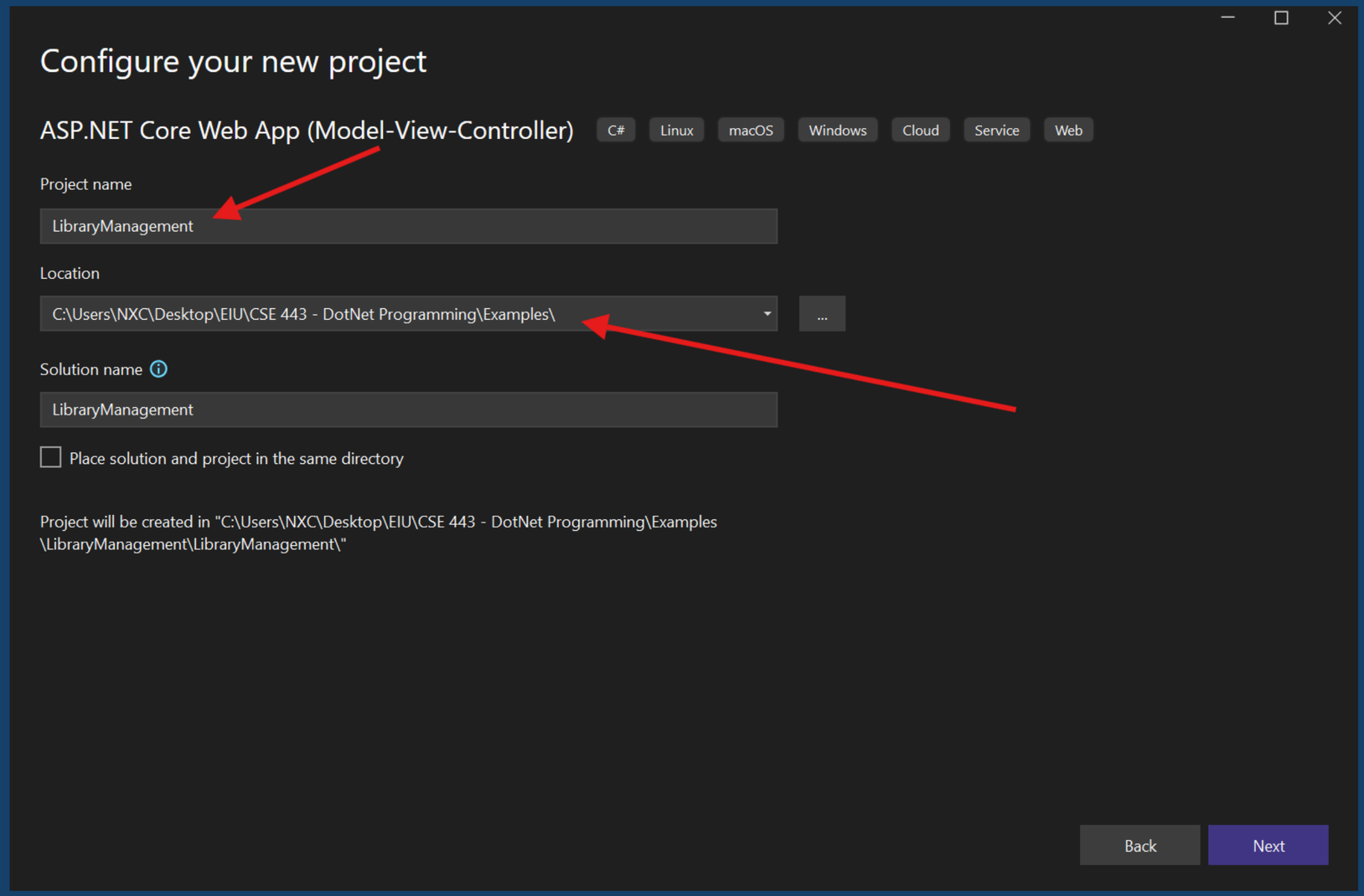
ASP.NET Core MVC

Create a web app:



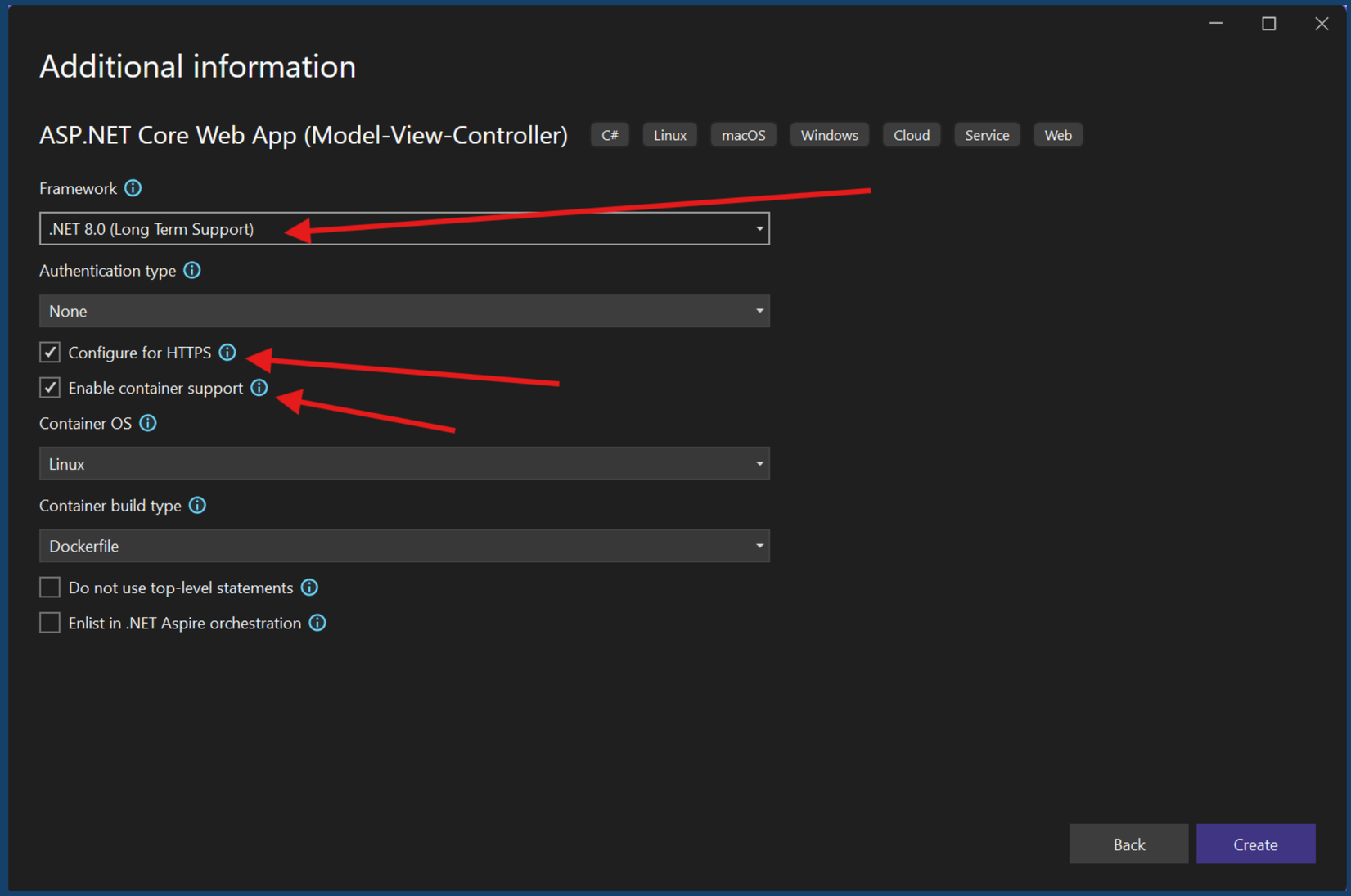
ASP.NET Core MVC

Create a web app:



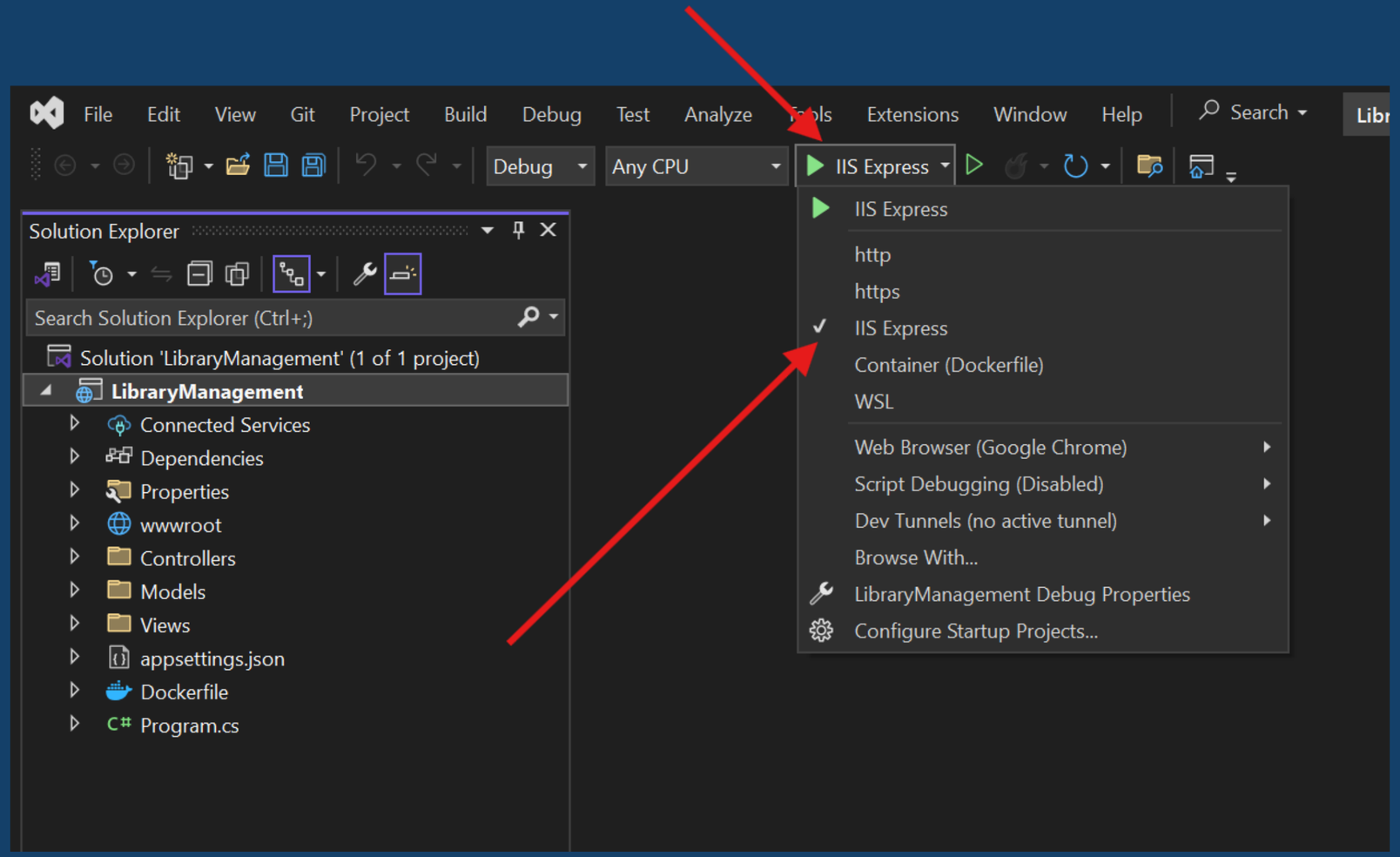
ASP.NET Core MVC

Create a web app:



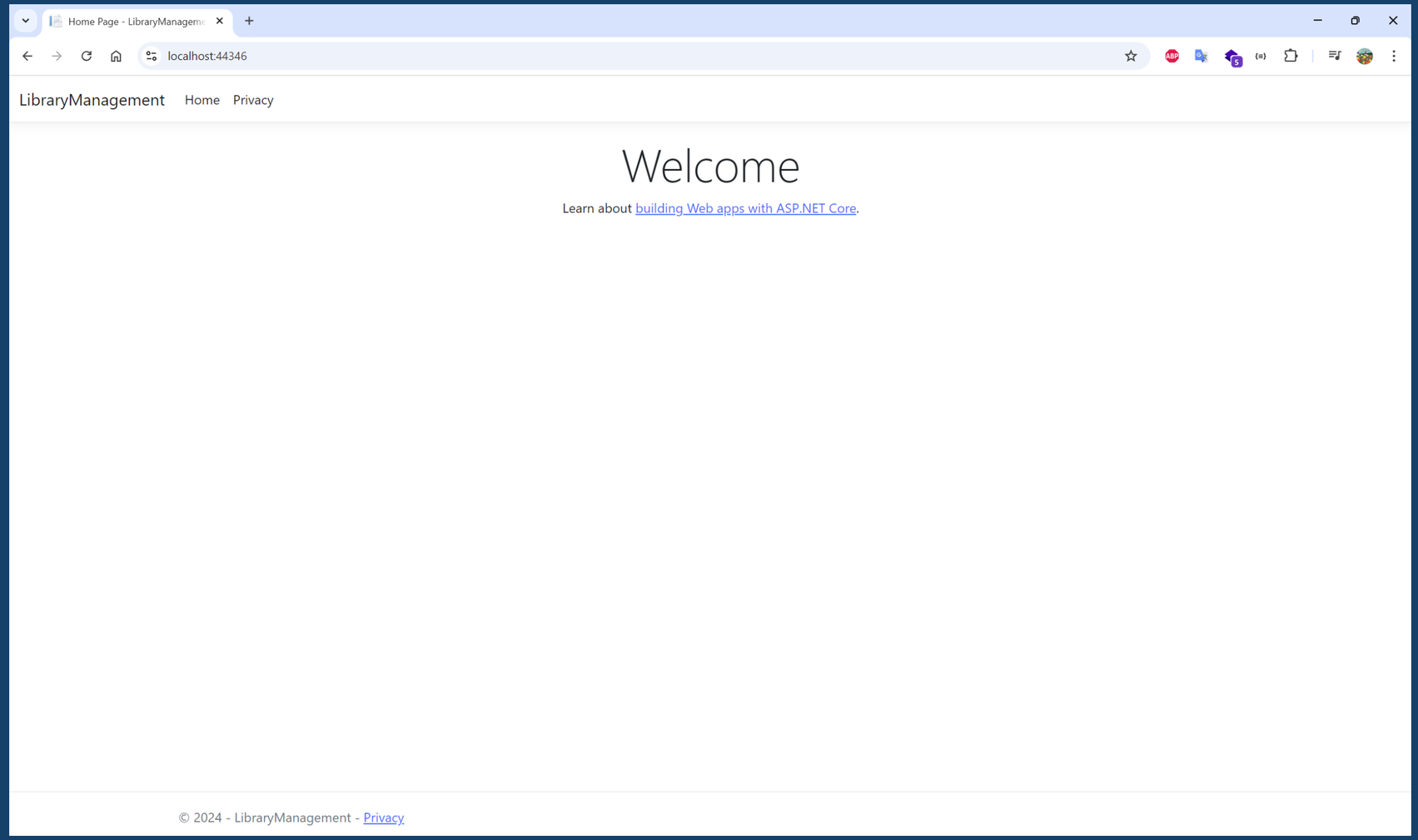
ASP.NET Core MVC

Run the app



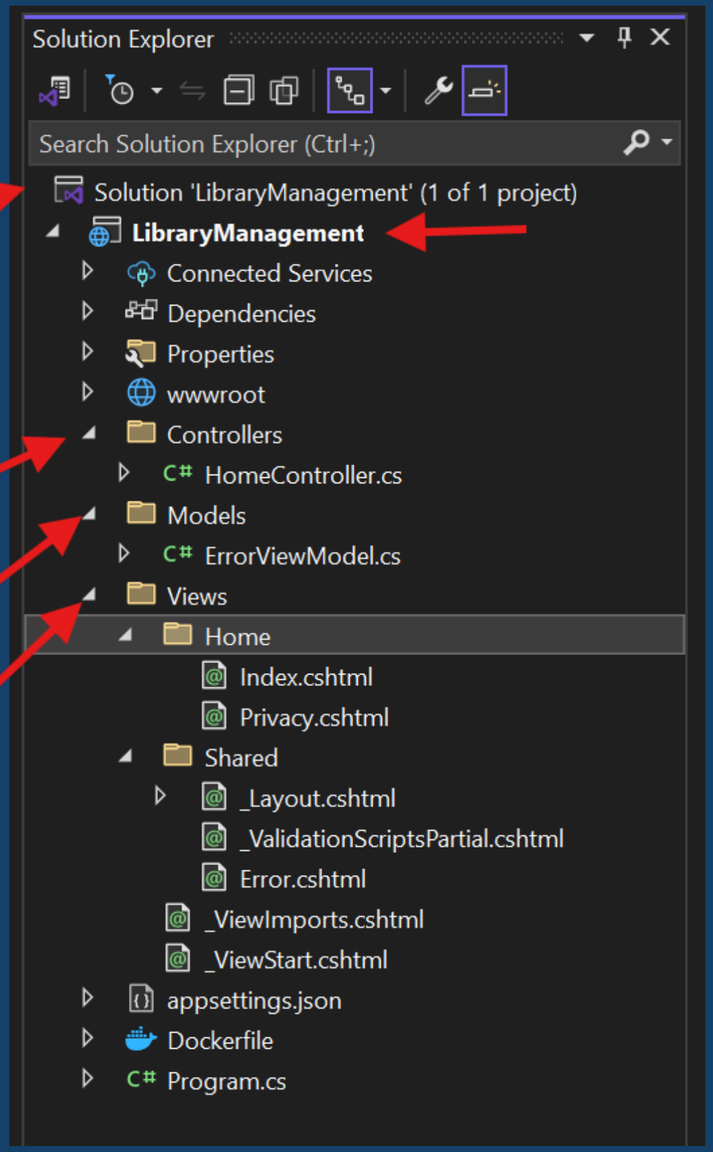
ASP.NET Core MVC

Run the app



ASP.NET Core MVC

Folder struct



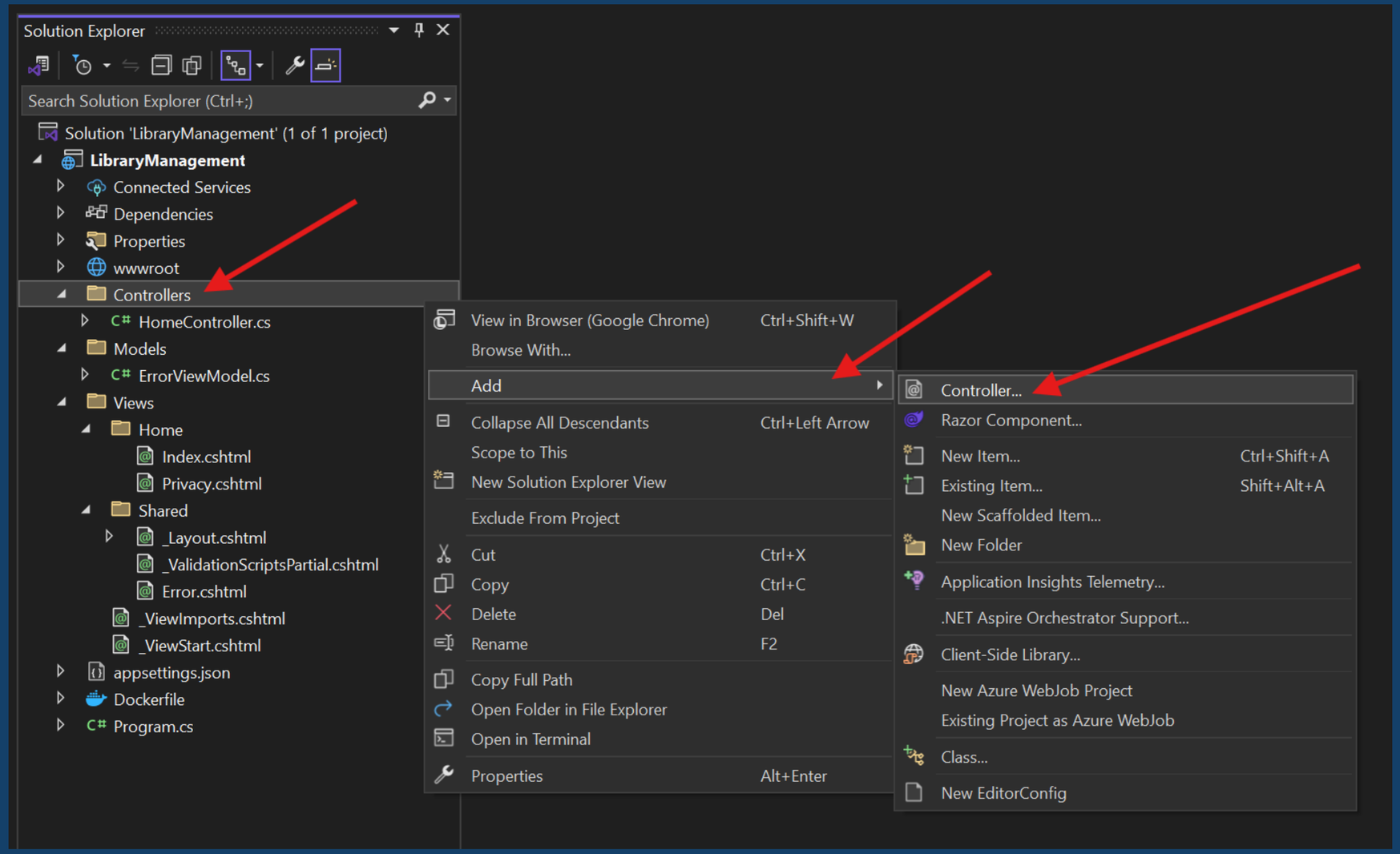
ASP.NET Core MVC

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.

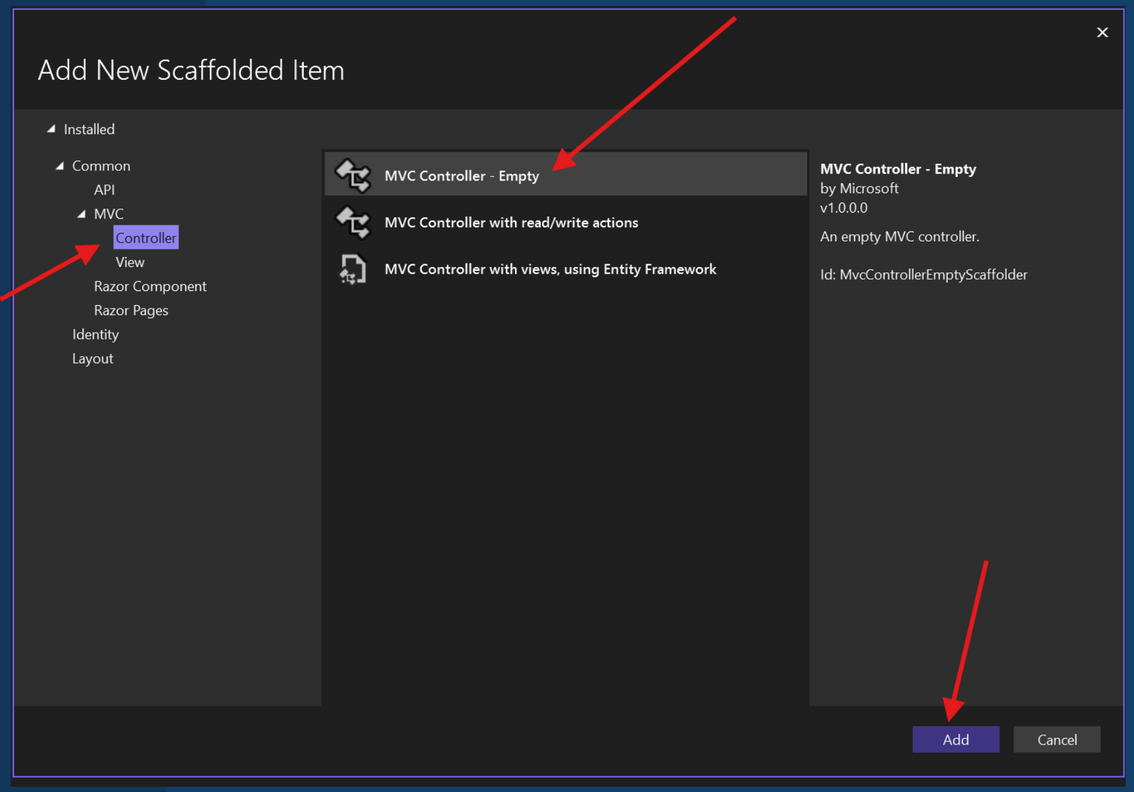
ASP.NET Core MVC

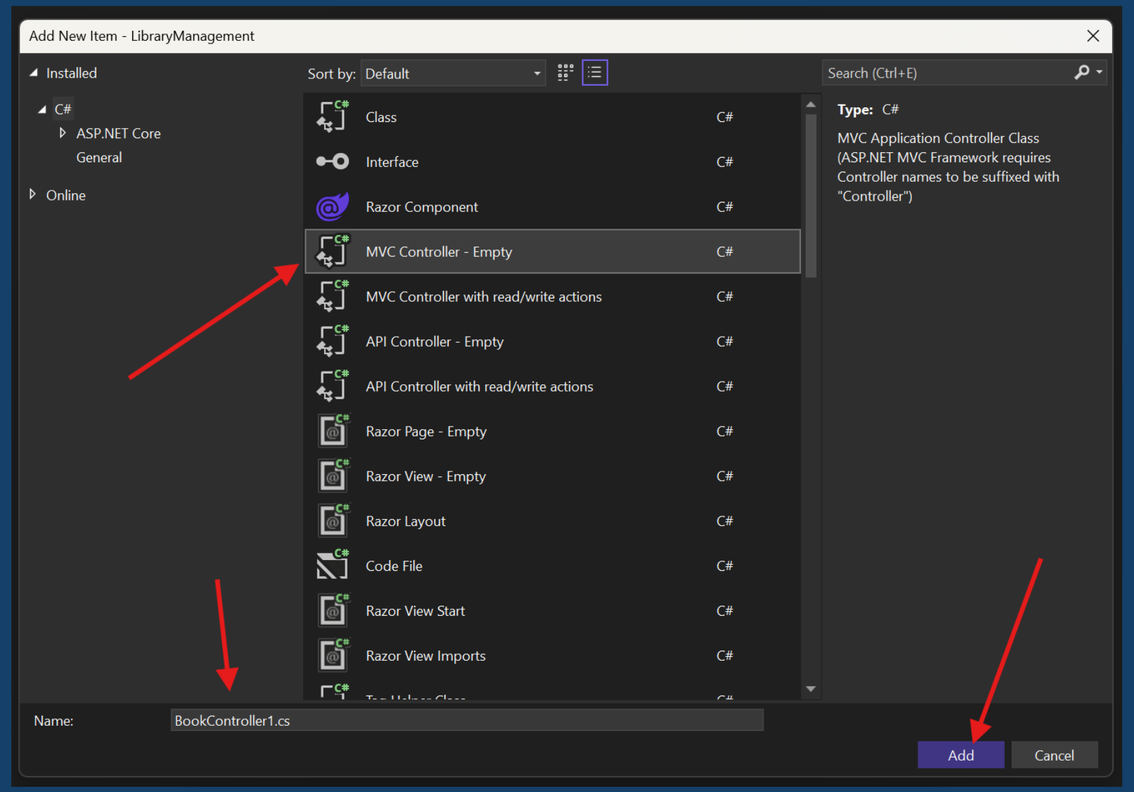
Controller - Add a controller



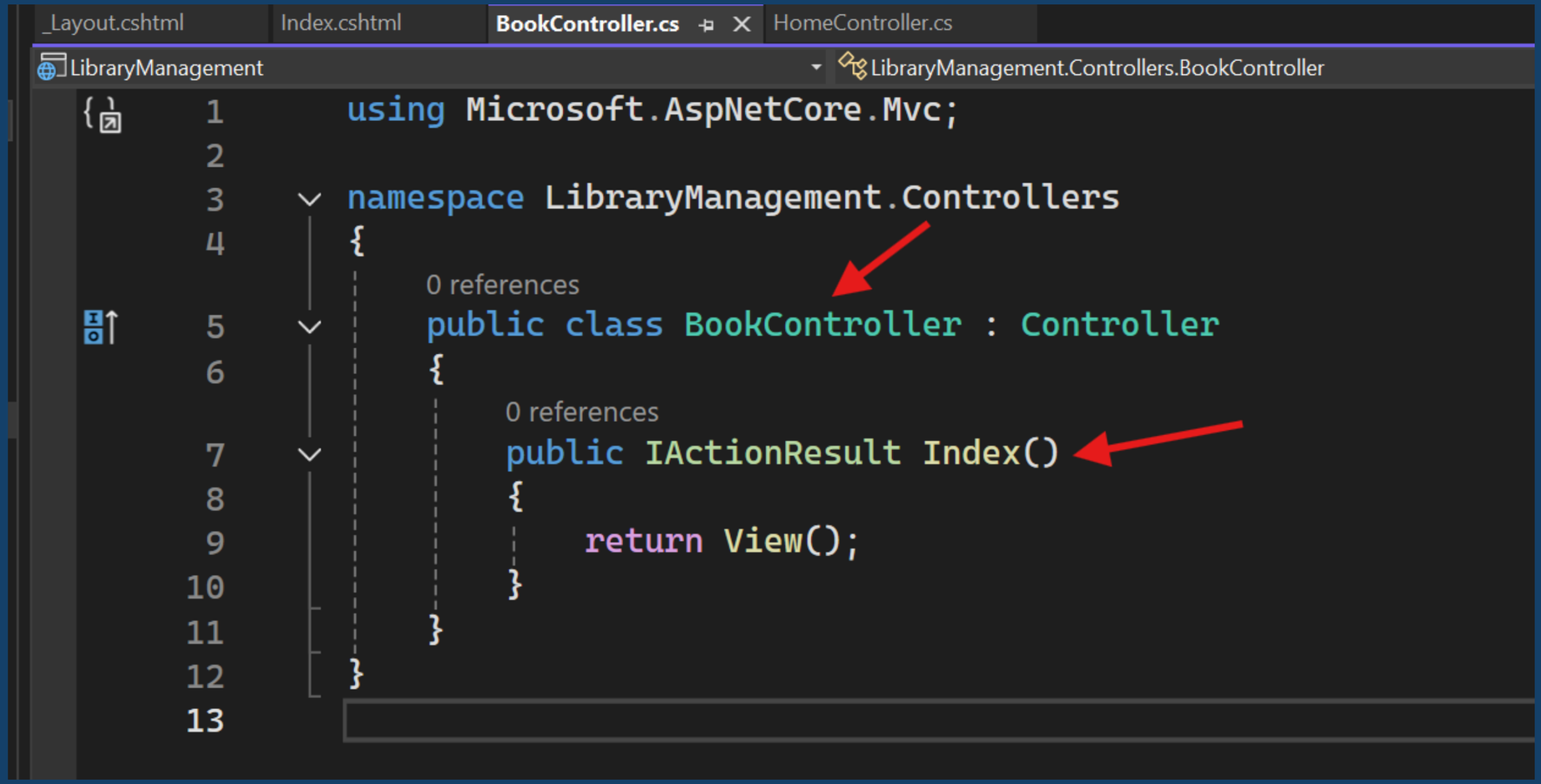
ASP.NET Core MVC

Controller - Add a controller

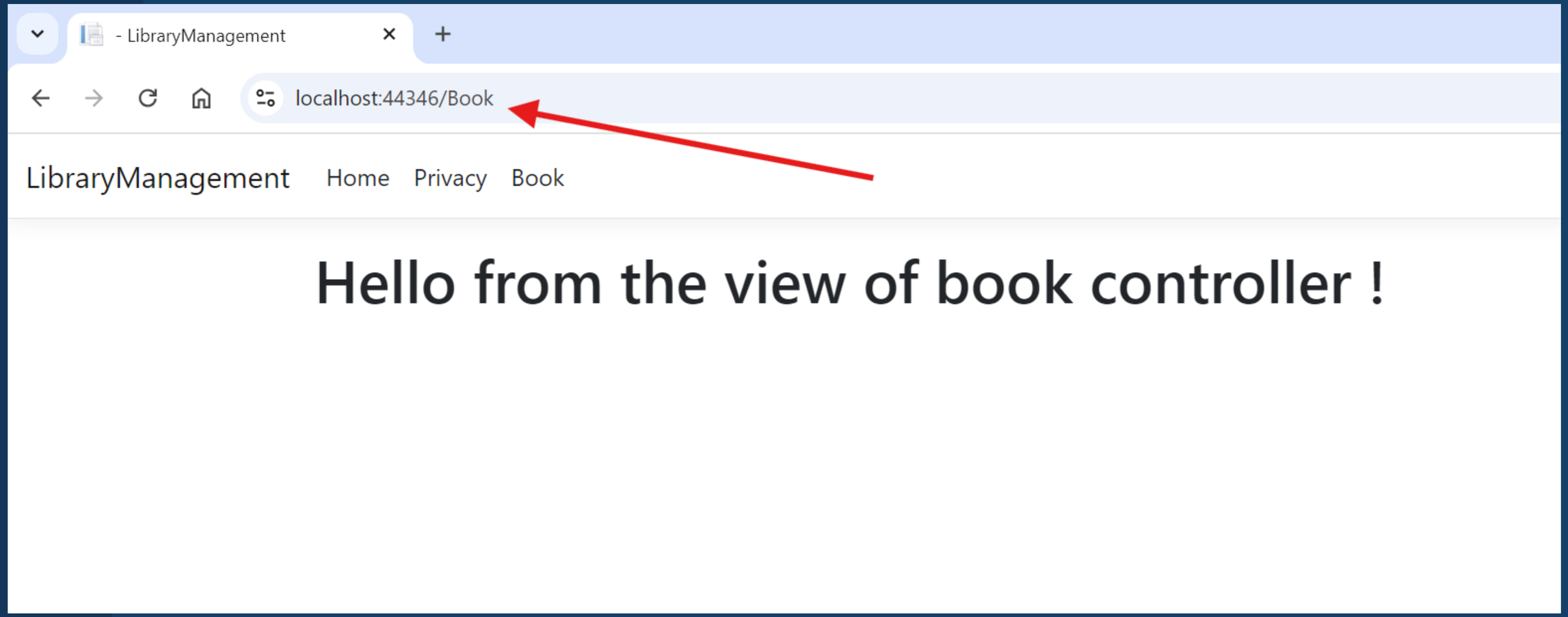




Controller – Index method



Controller – view from controller



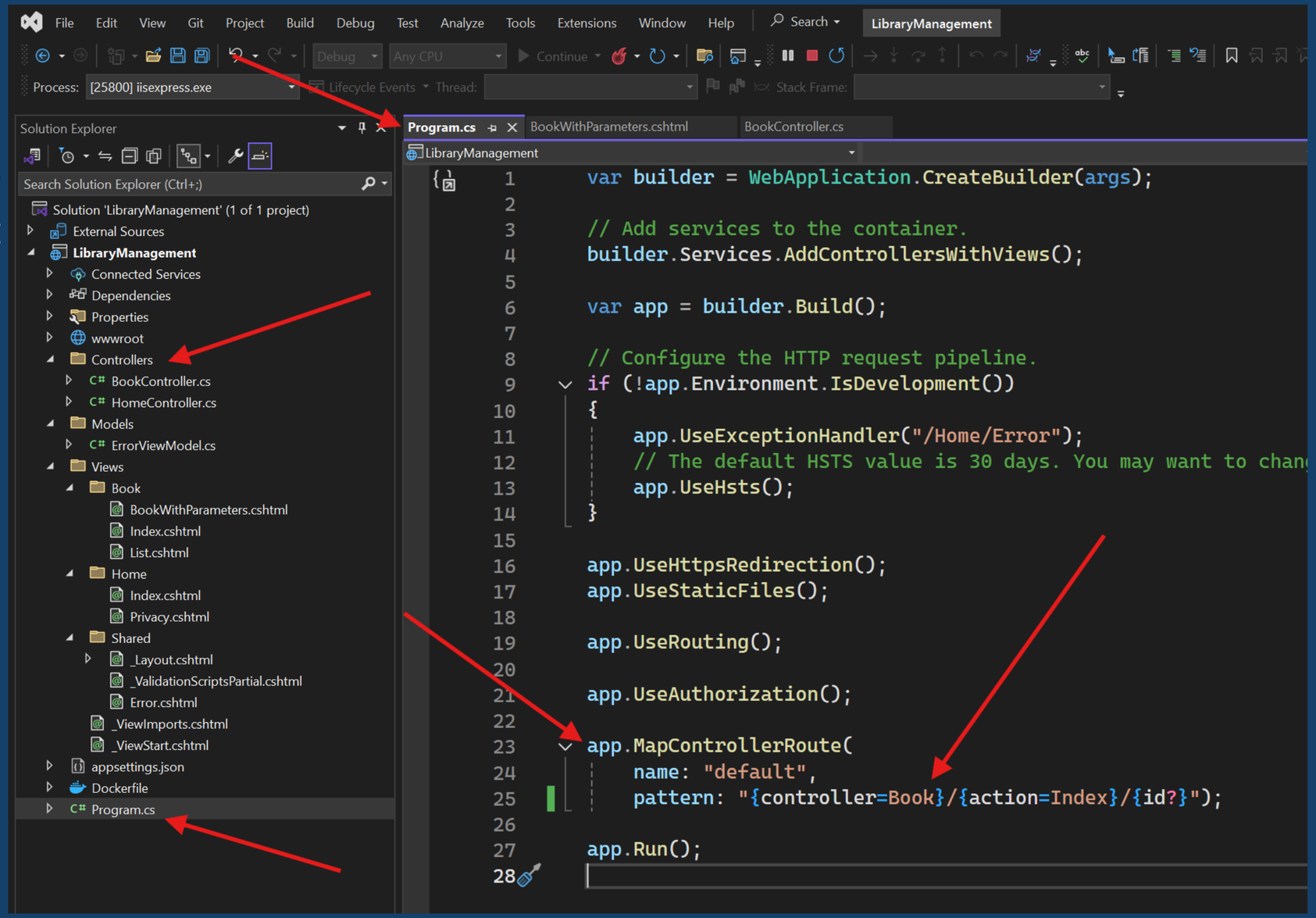
Controller – HTTP Endpoint

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.

Controller – Default Endpoint

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



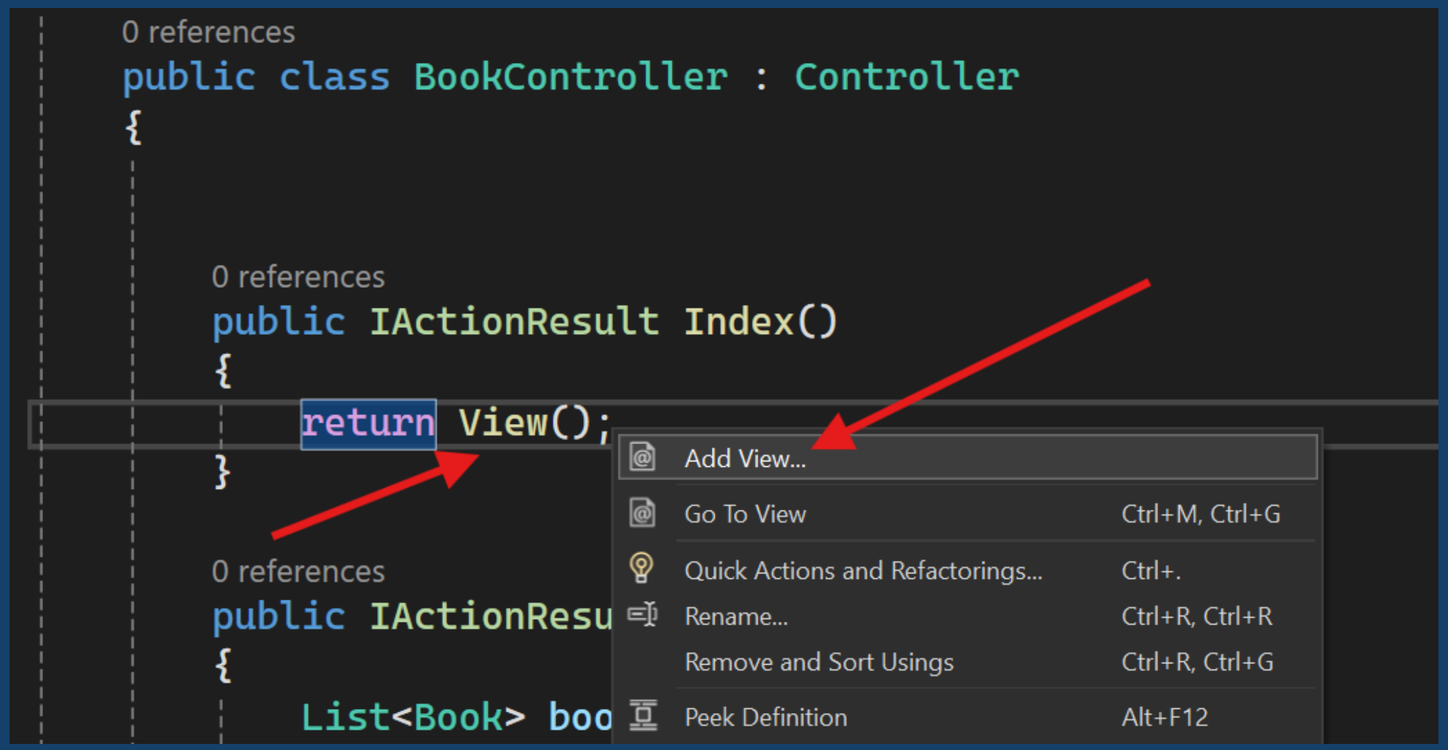
View

ASP.NET Core MVC

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

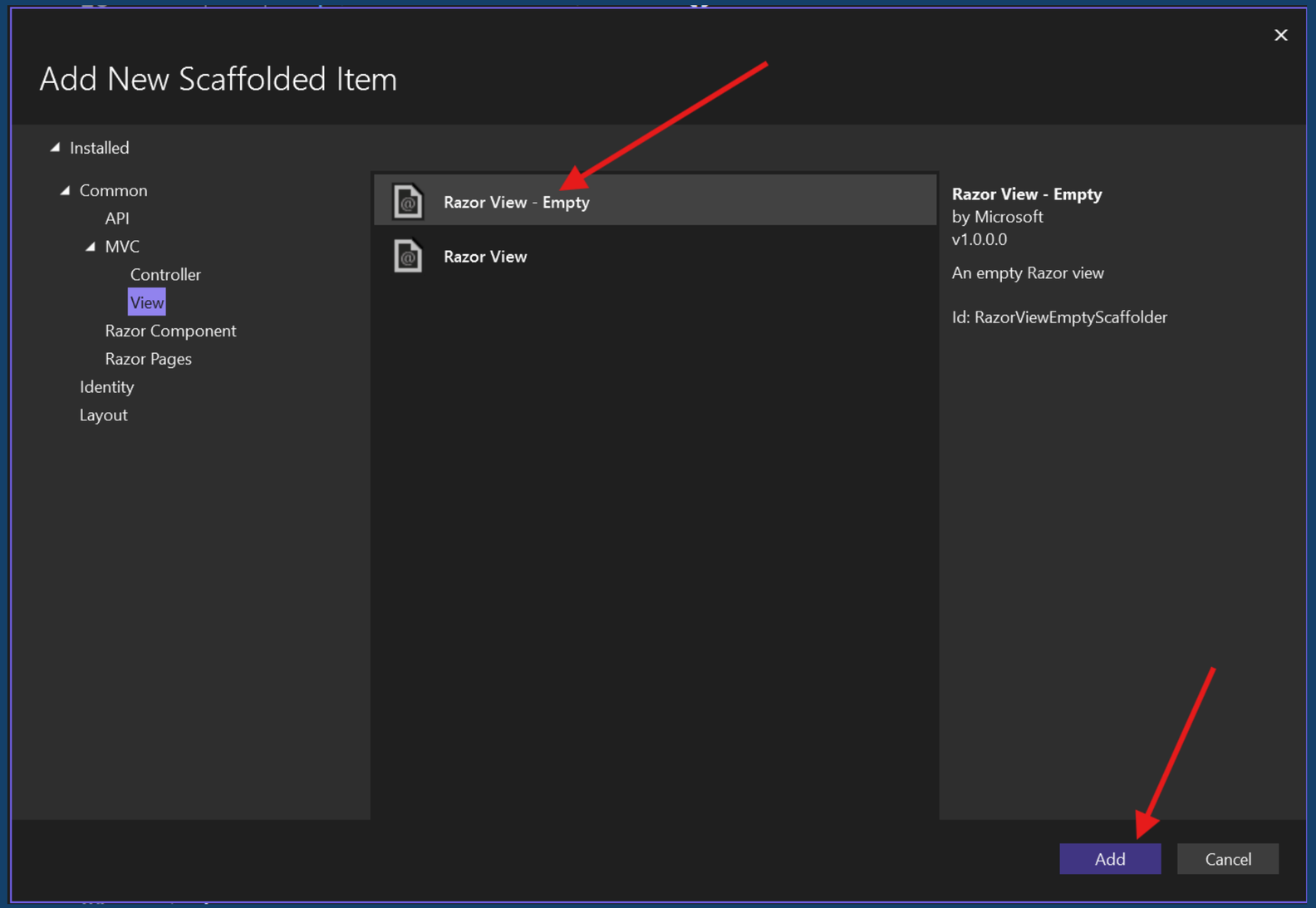
View templates are created using Razor. Razor-based view templates:

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



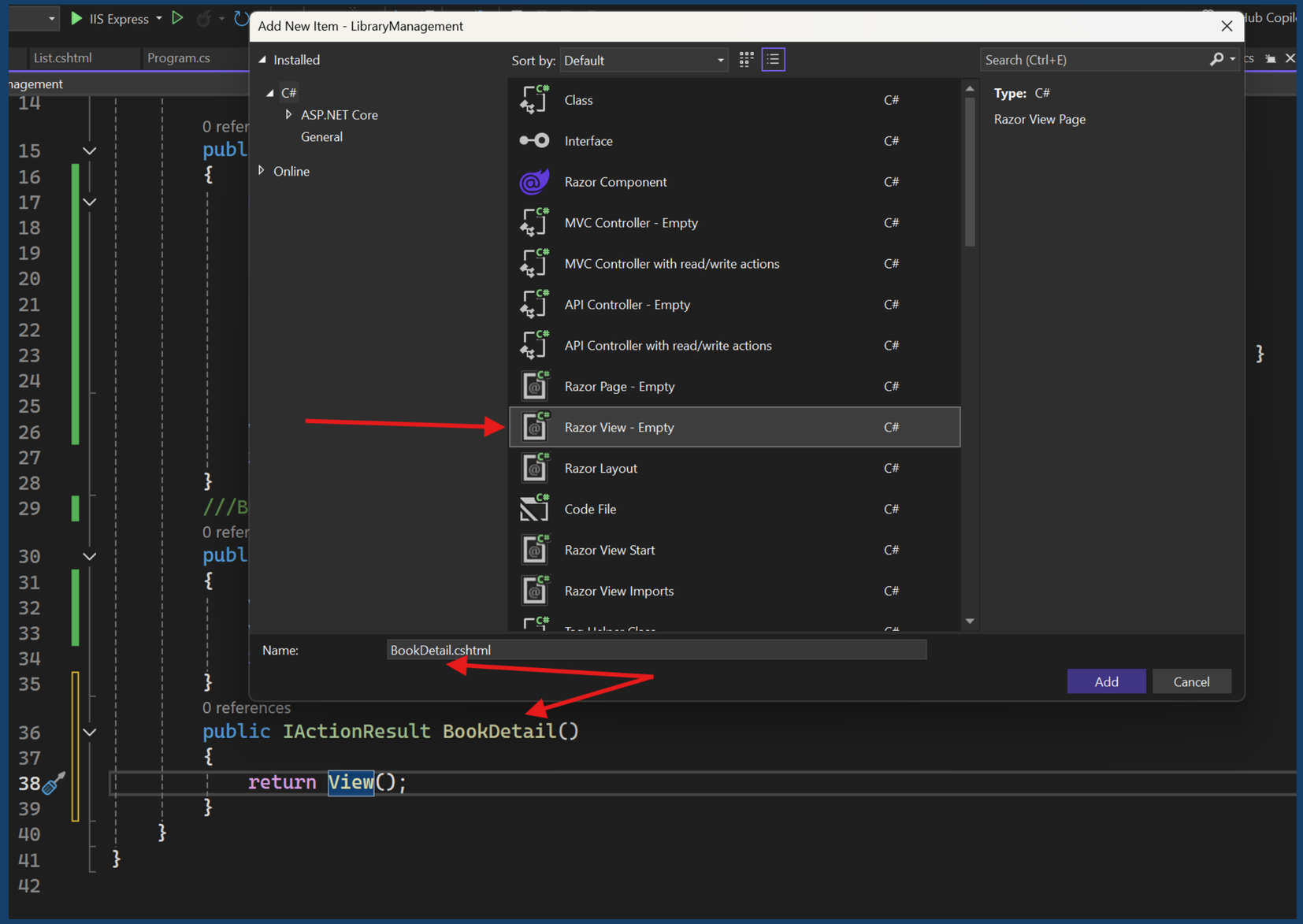
ASP.NET Core MVC

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



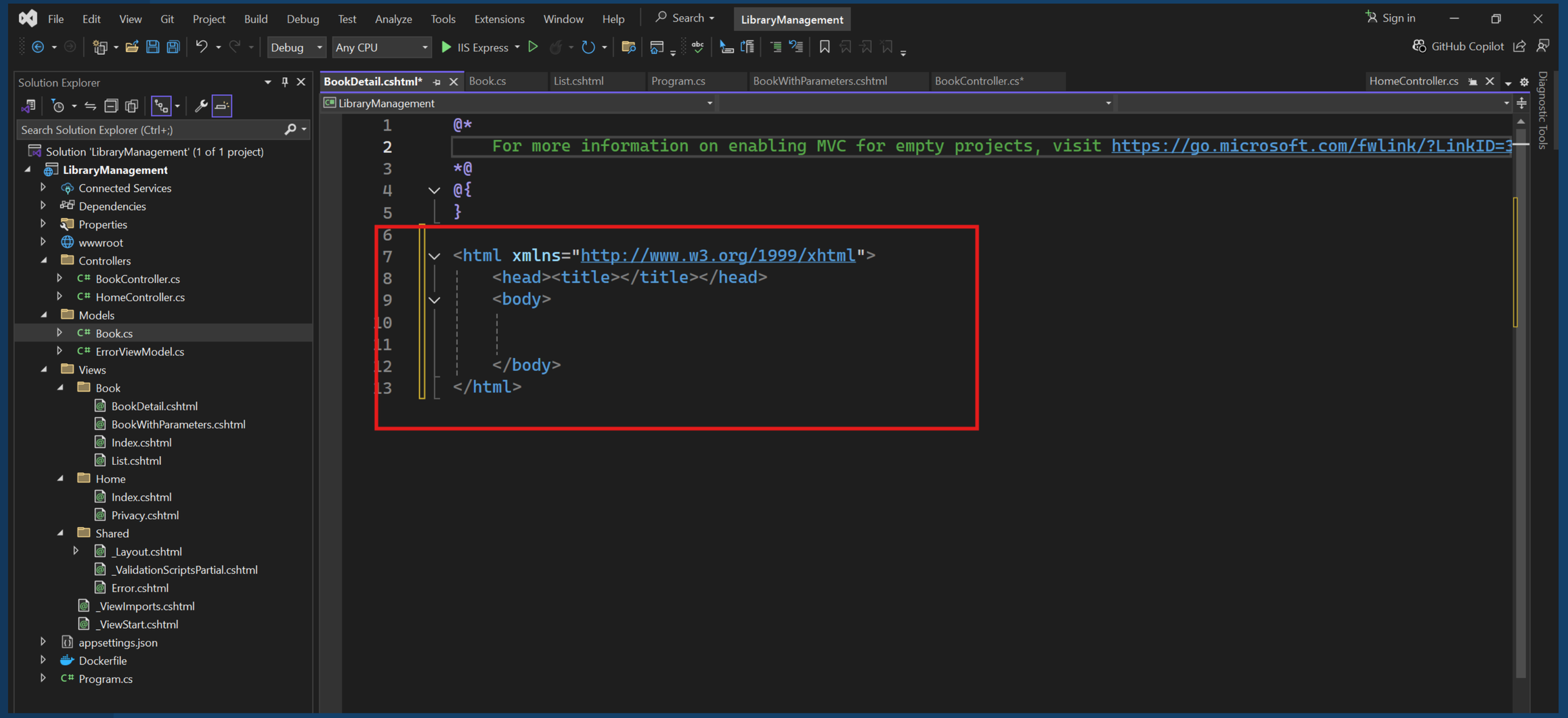
ASP.NET Core MVC

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



ASP.NET Core MVC

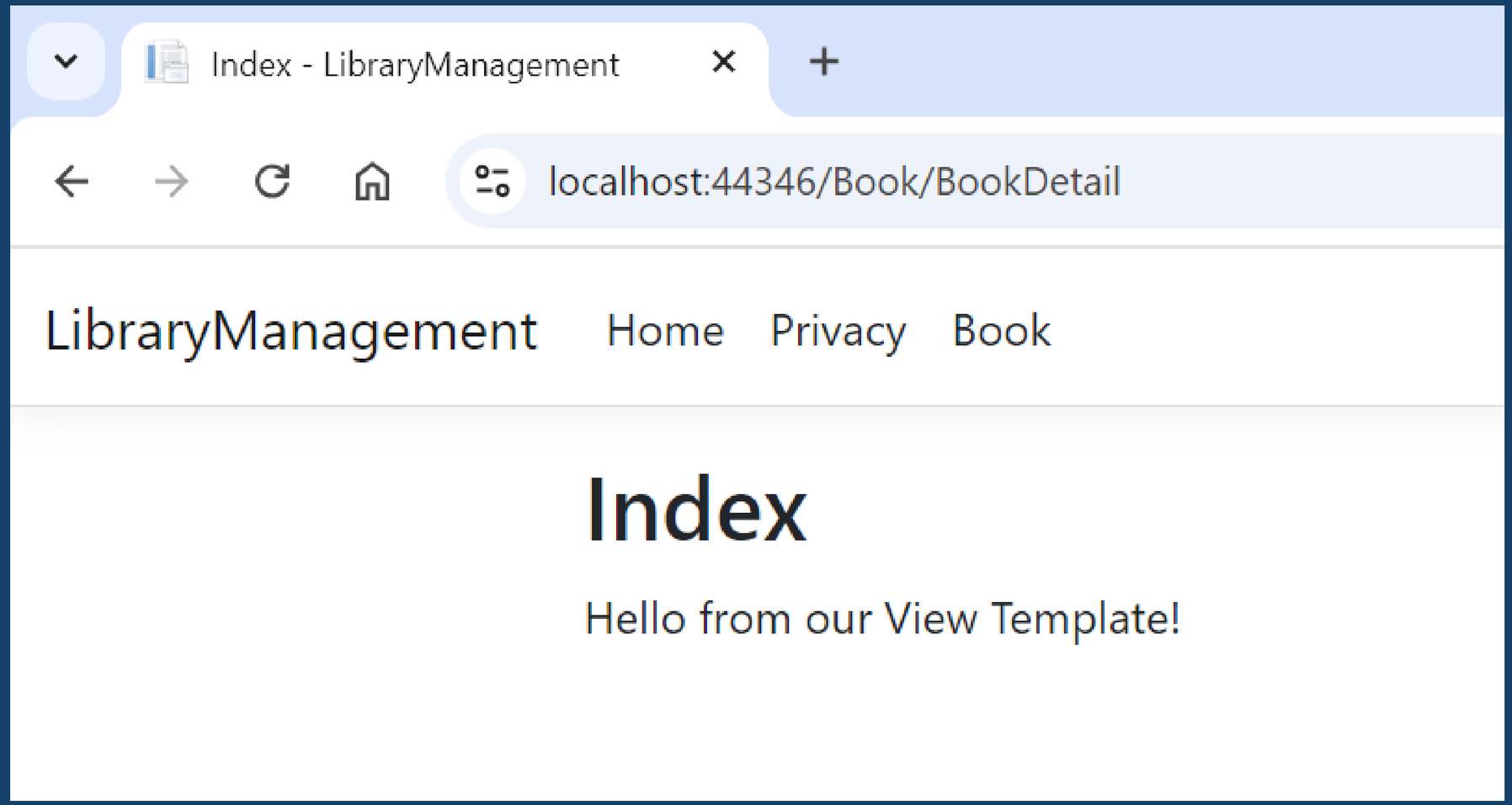
View – The Razor view after create success



ASP.NET Core MVC

View – Show in the website

Navigate to https://localhost:{PORT}/{Controller}: // check the path in your project



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

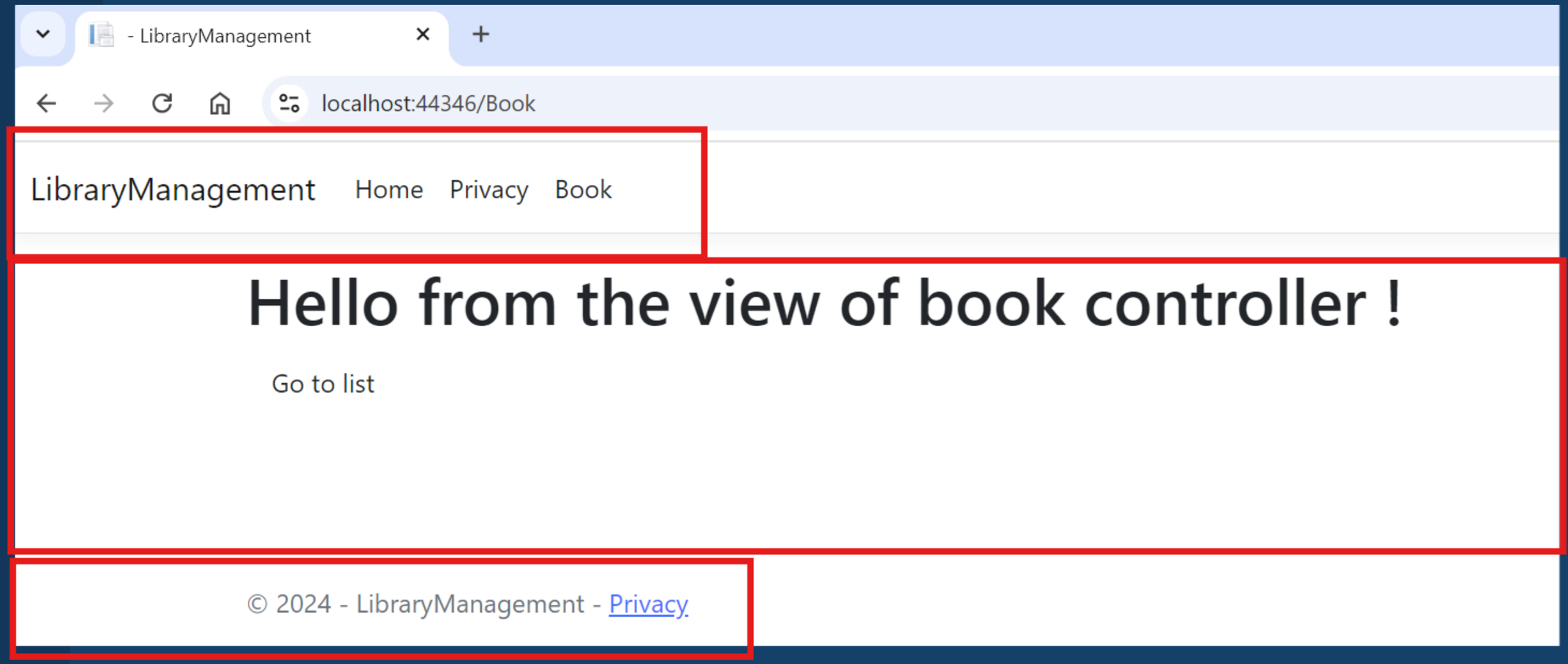
Select the menu links LibraryManagement, Home, and Privacy. Each page shows the same menu layout. The menu layout is implemented in the Views/Shared/_Layout.cshtml file.

Open the Views/Shared/_Layout.cshtml file.

Layout templates allow:

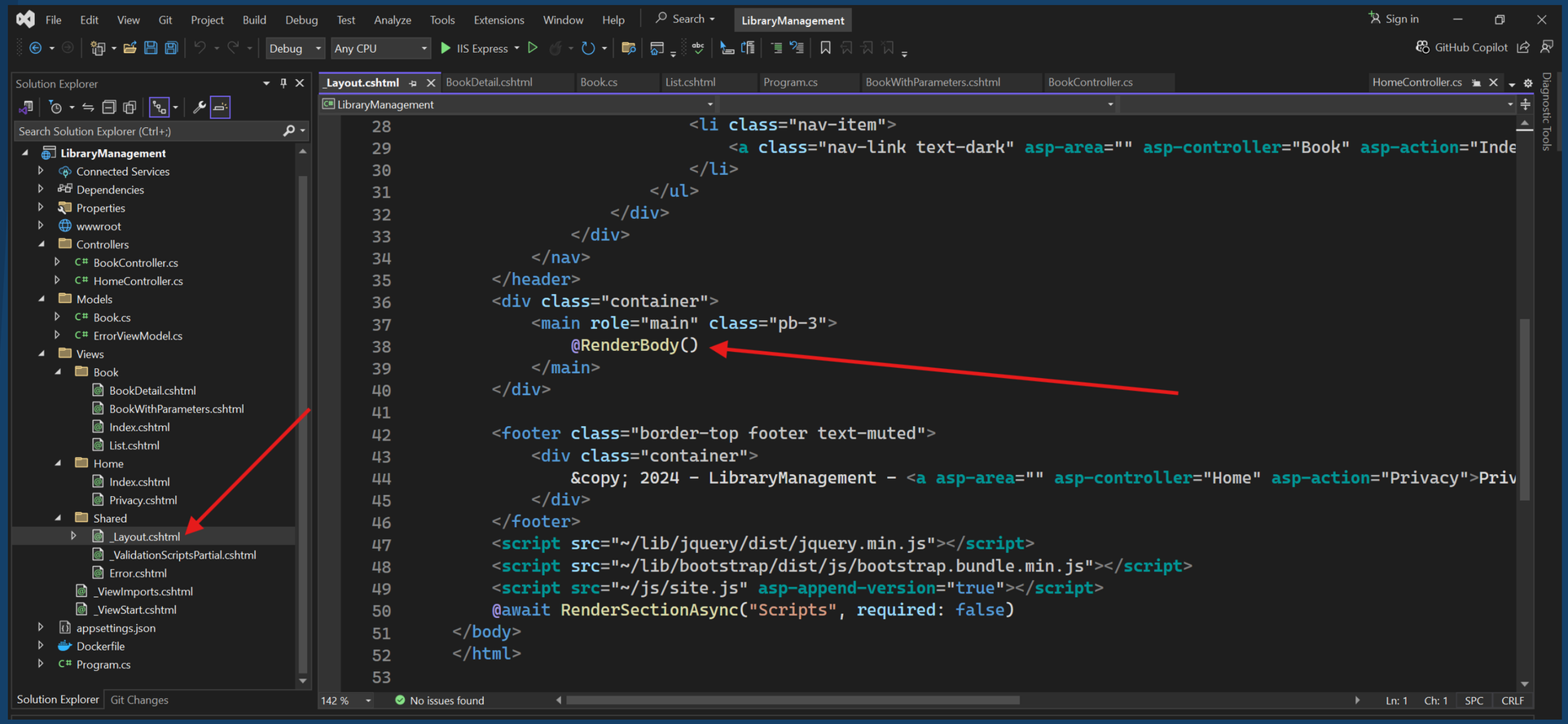
- Specifying the HTML container layout of a site in one place.
- Applying the HTML container layout across multiple pages in the site.

View – Change views and layout pages



View – Change views and layout pages

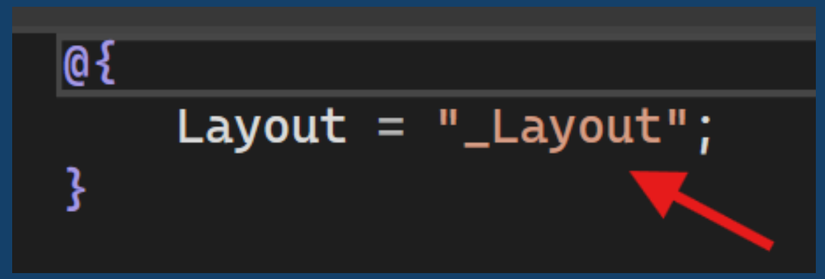
RenderBody is a placeholder where all the view-specific pages you create show up, wrapped in the layout page. For example, if you select the Privacy link, the Views/Home/Privacy.cshtml view is rendered inside the RenderBody method.



View – Default layout

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

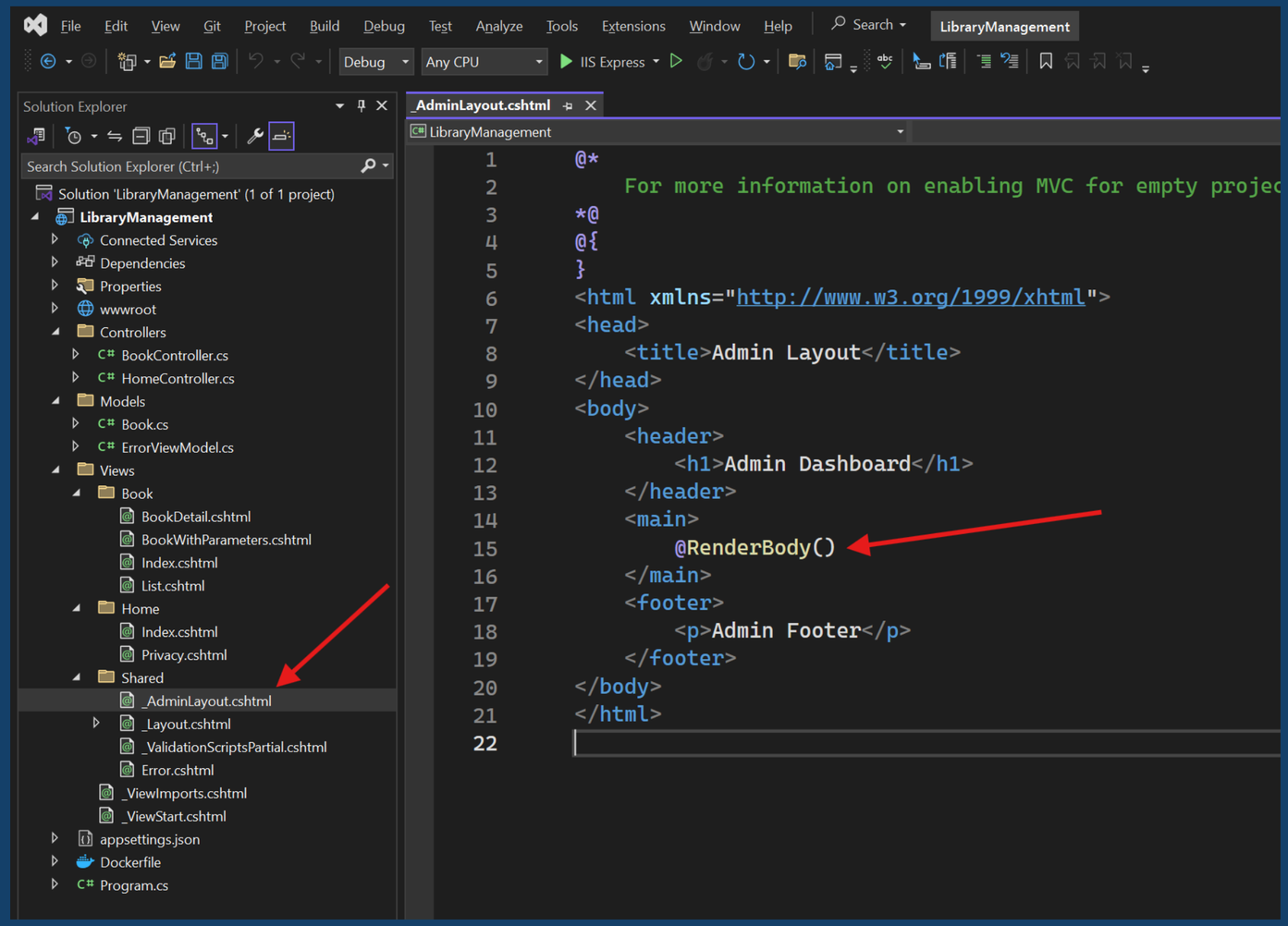
The Layout property can be used to set a different layout view, or set it to null so no layout file will be used.



ASP.NET Core MVC

View – Change the layout for each view

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



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

```
1   @*
2       For more information on enabling MVC for empty projects, vis
3   *@
4   @{
5   }
6   @{
7       Layout = "_AdminLayout";
8   }
9   @{
10      ViewData["Title"] = "Index";
11  }
12
13
14  <html xmlns="http://www.w3.org/1999/xhtml">
15      <head><title></title></head>
16      <body>
17
18      <h2>Index</h2>
19
20      <p>Hello from our View Template!</p>
21      </body>
22  </html>
```

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.

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();
}
```

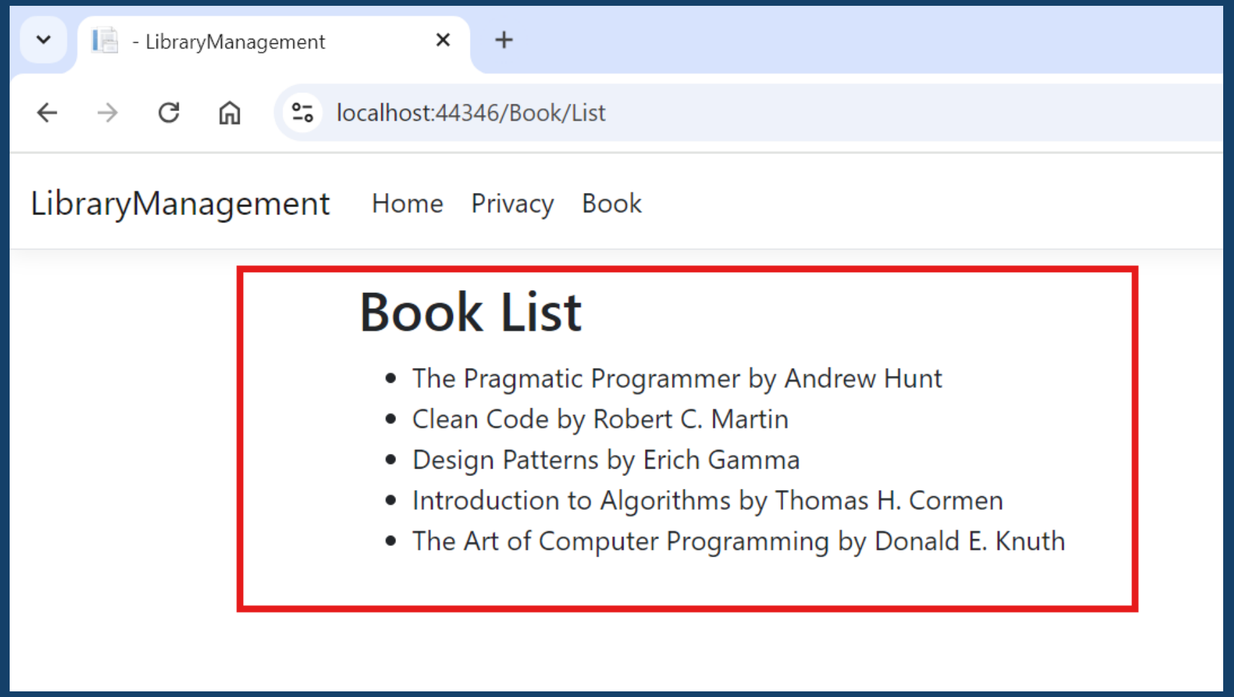
View – Passing Data from the Controller to the View

- Show the data get from controller to the view

```
ement
    @*
        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>
```



Model

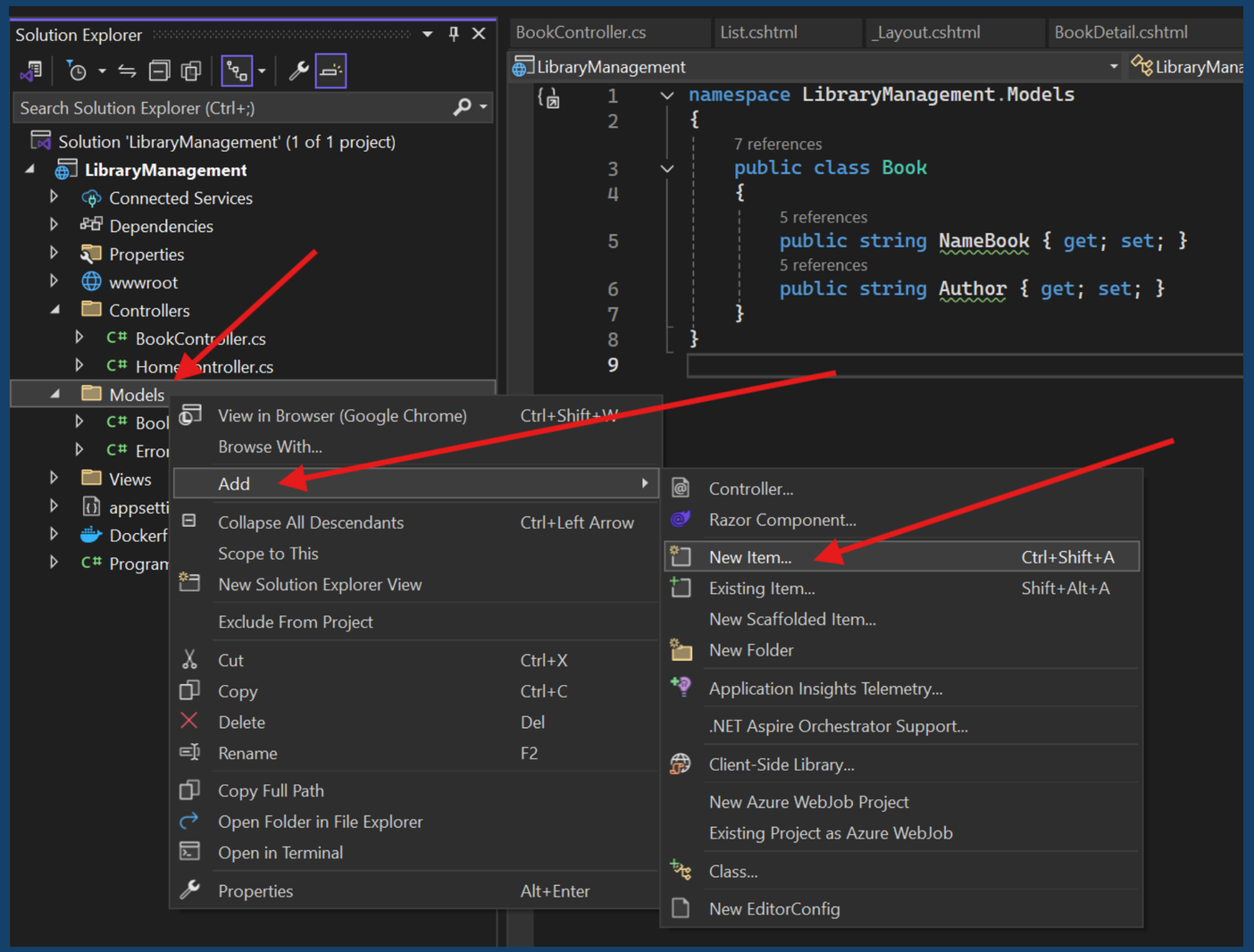
Model

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

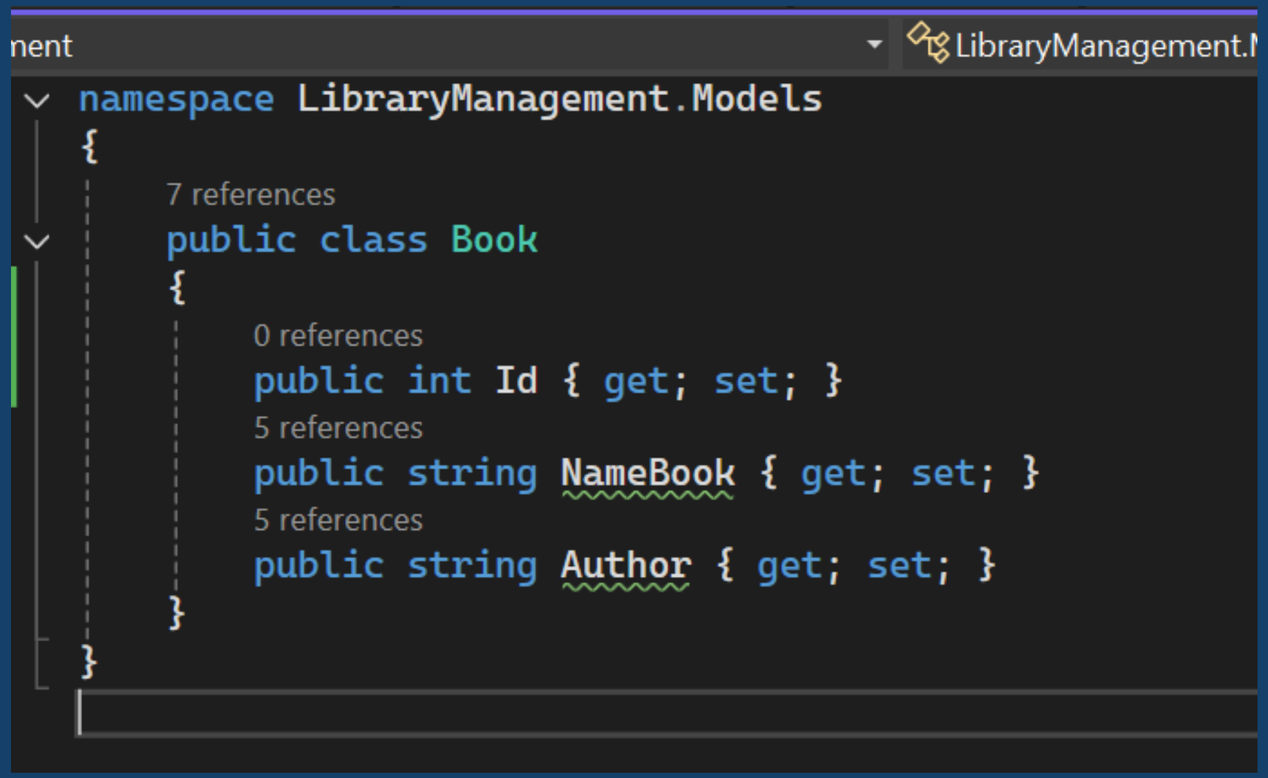


Model – Add a model to an ASP.NET Core MVC app

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

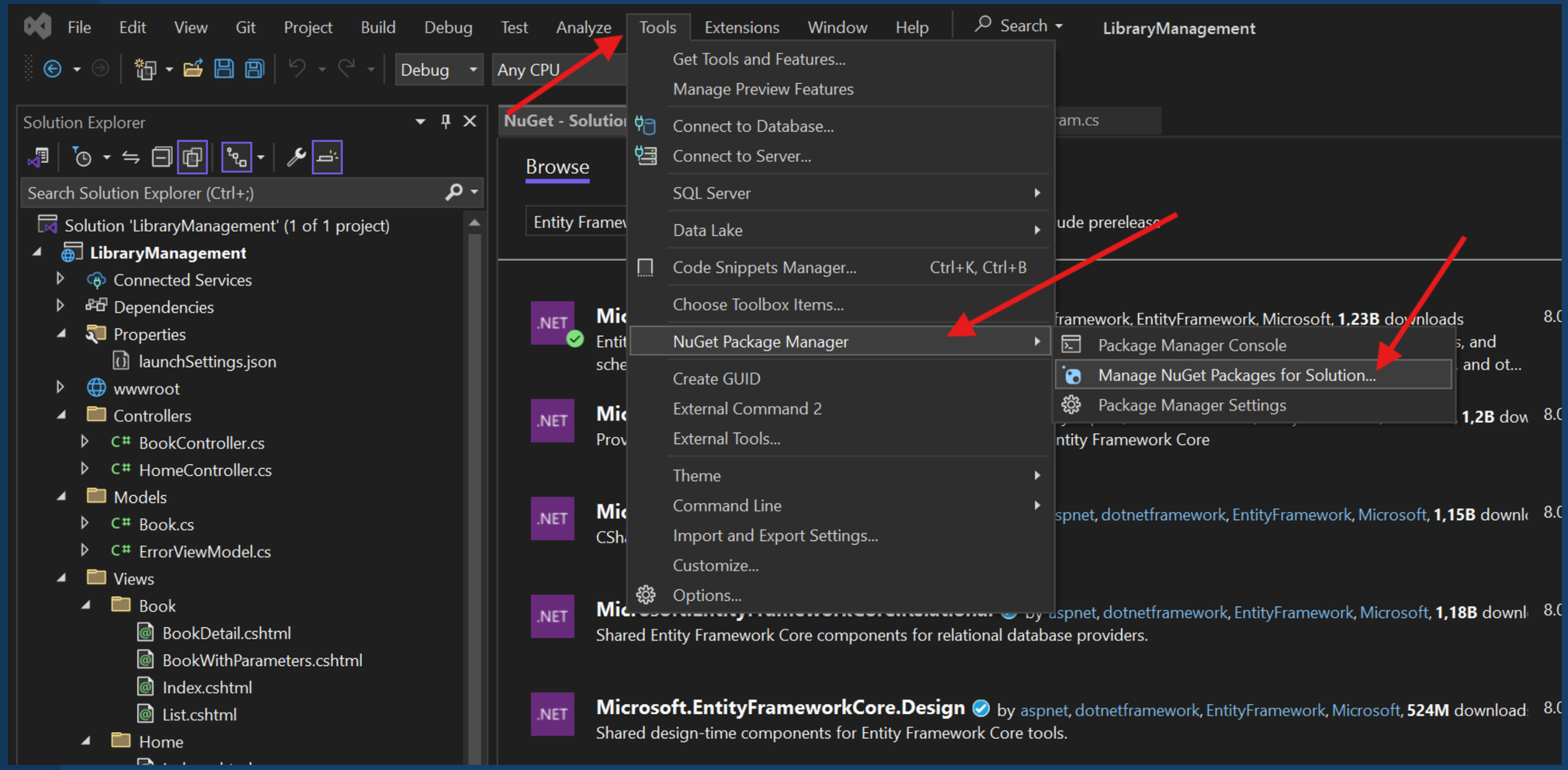
The DataType attribute on ReleaseDate specifies the type of the data (Date). With this attribute:

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



ASP.NET Core MVC

Model – Entity Framework

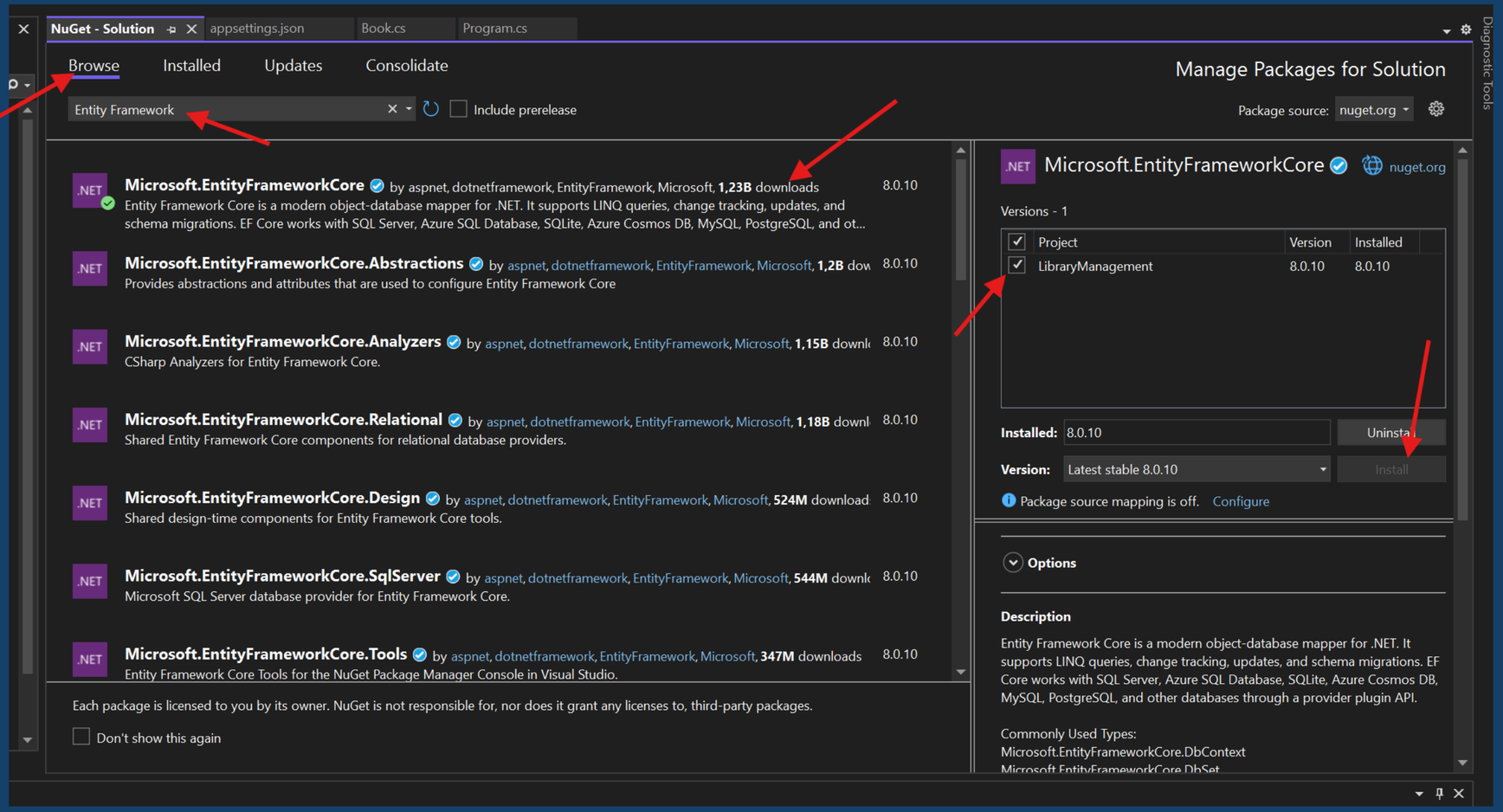


ASP.NET Core MVC

Model – Entity Framework

Microsoft.EntityFrameworkCore.SqlServer

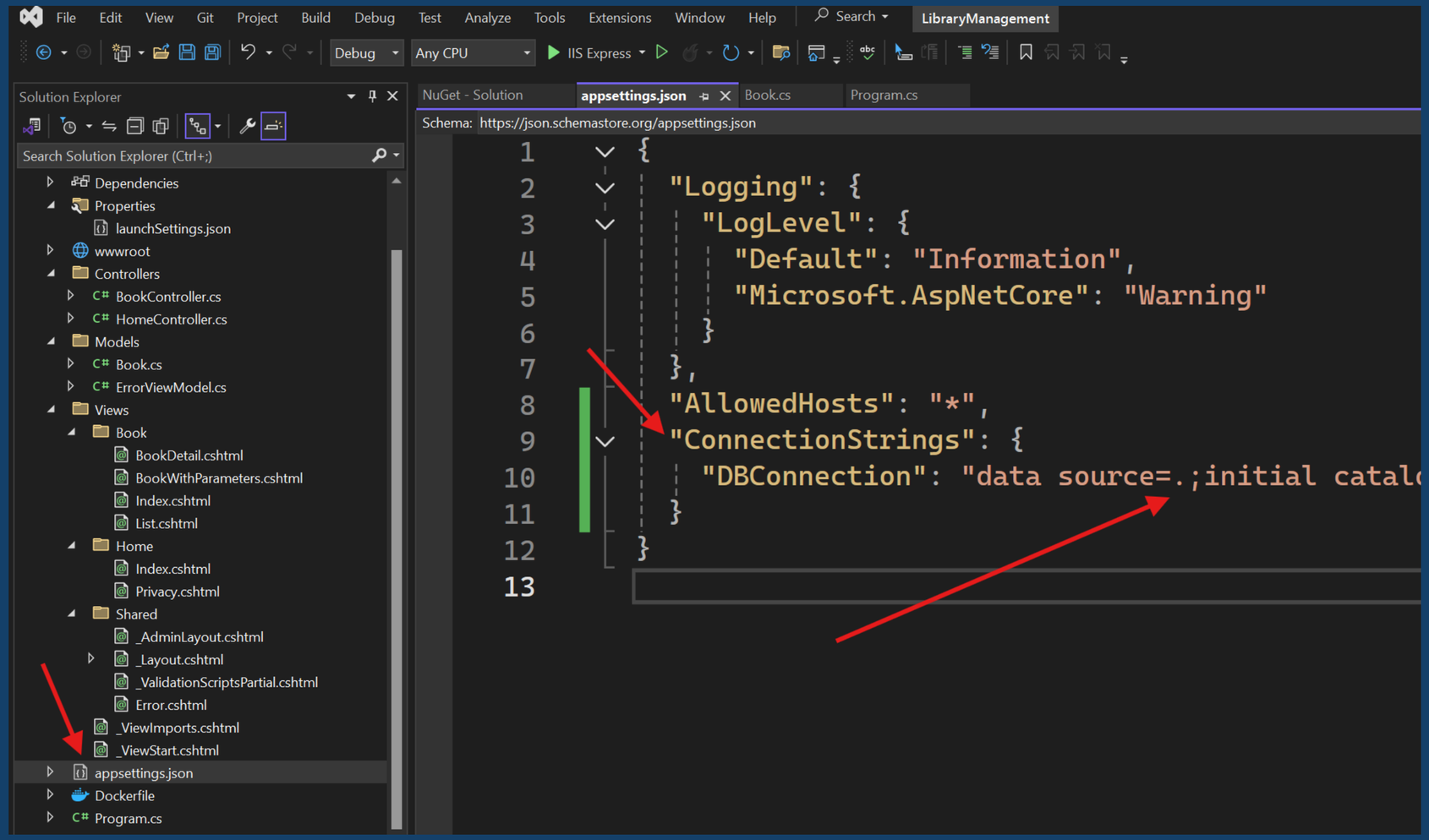
Microsoft.EntityFrameworkCore



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

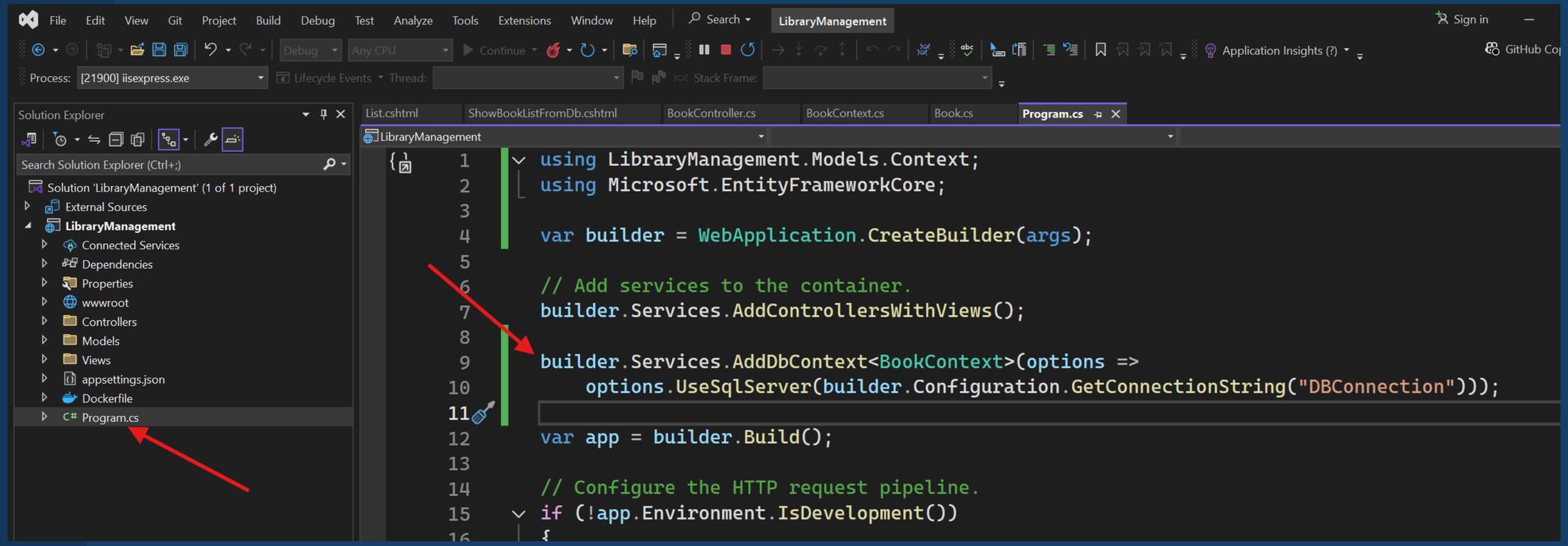


ASP.NET Core MVC

Model – Dependency injection

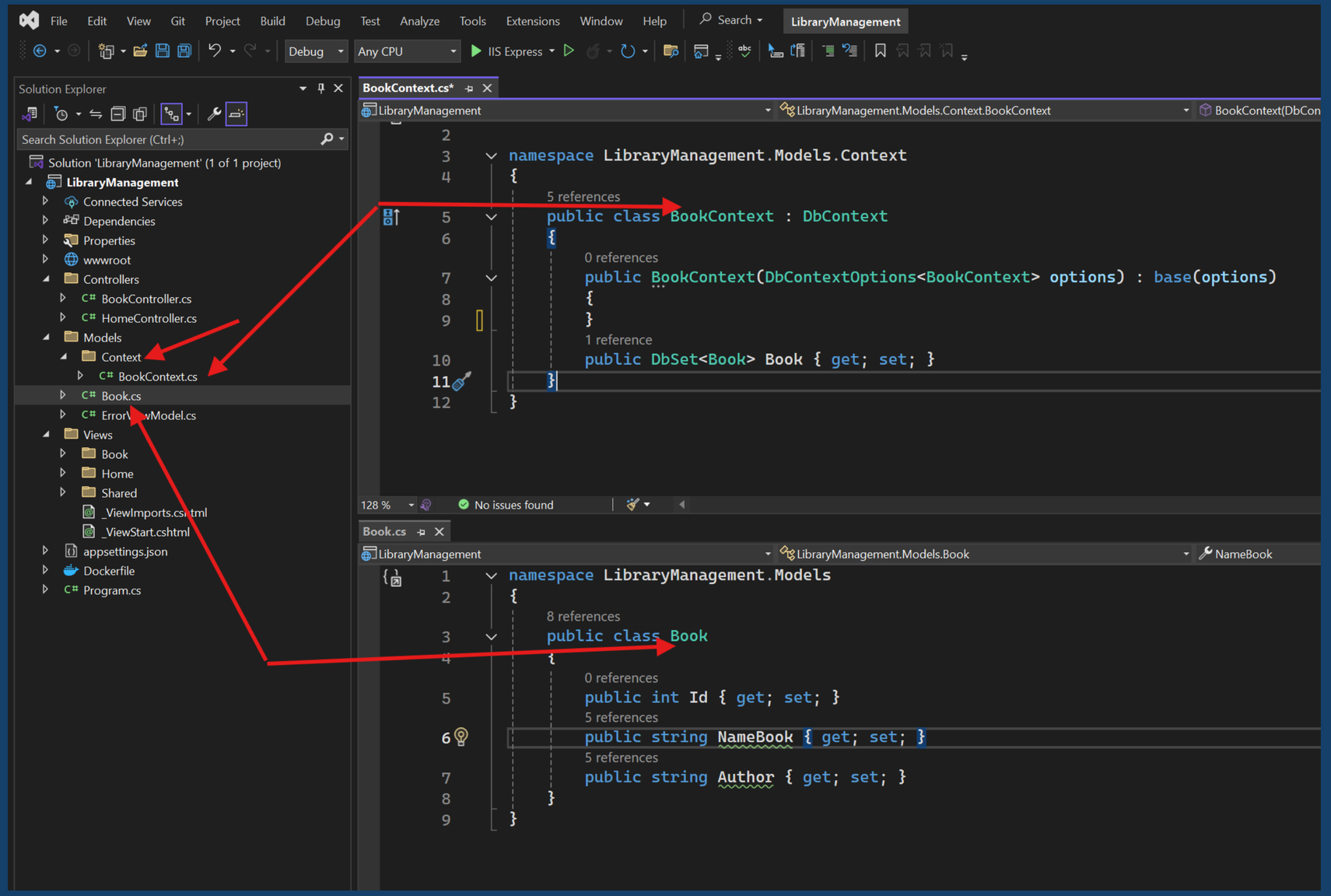
- ASP.NET Core is built with dependency injection (DI). Services, such as the database context, are registered with DI in Program.cs. These services are provided to components that require them via constructor parameters.
- In the Controllers/MoviesController.cs file, the constructor uses Dependency 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:

Model – Dependency injection

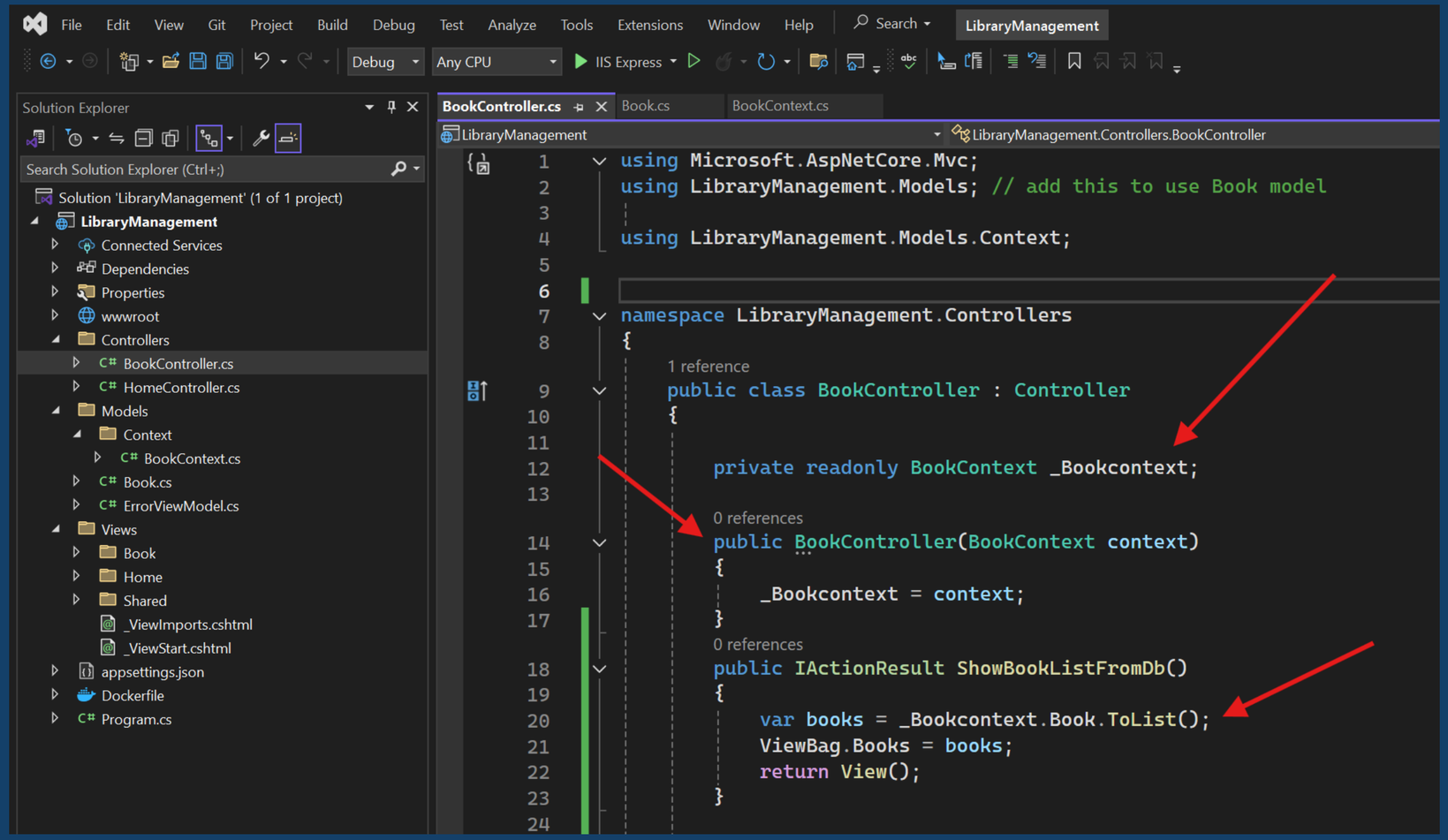


ASP.NET Core MVC

Model – Dbcontext

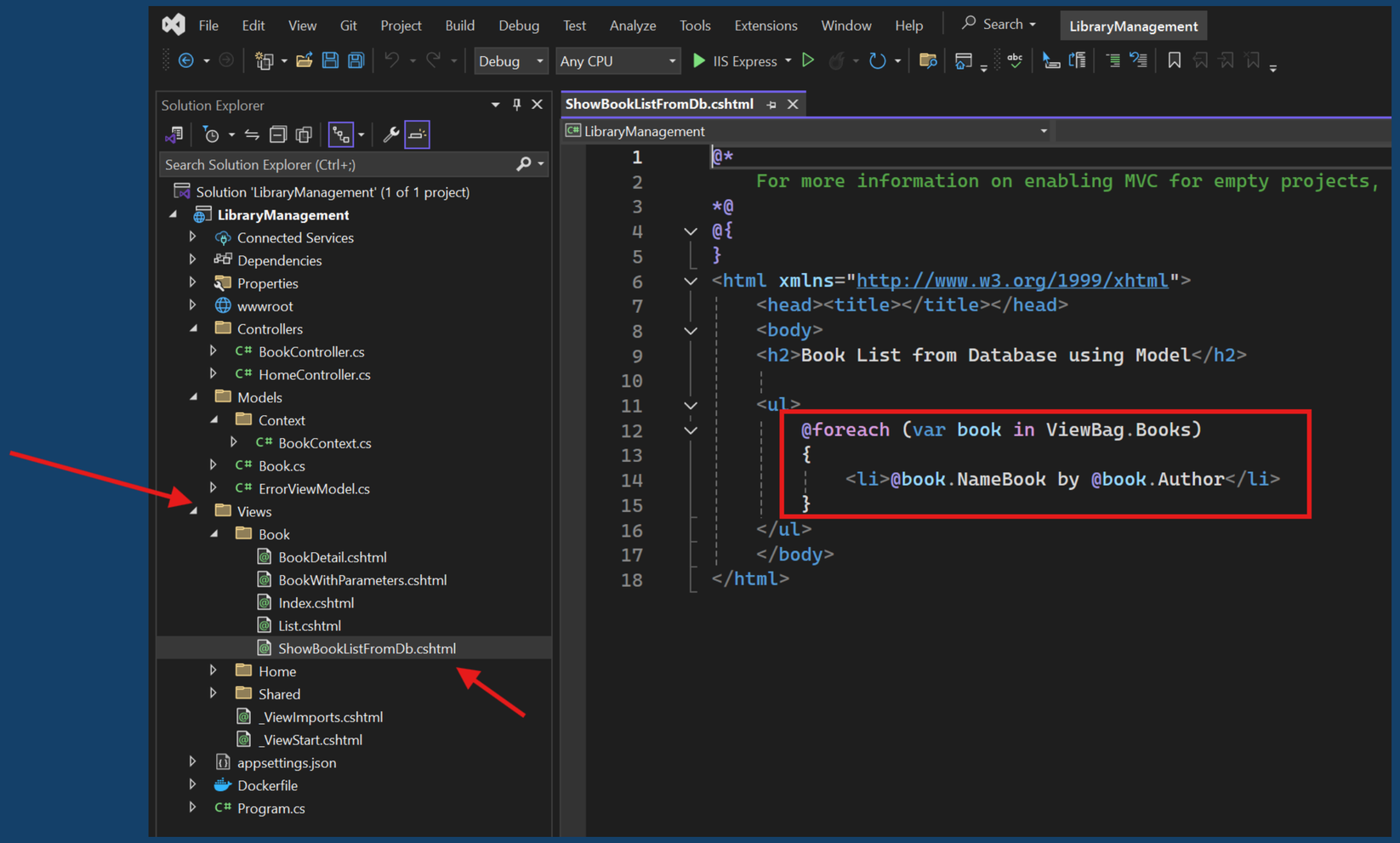


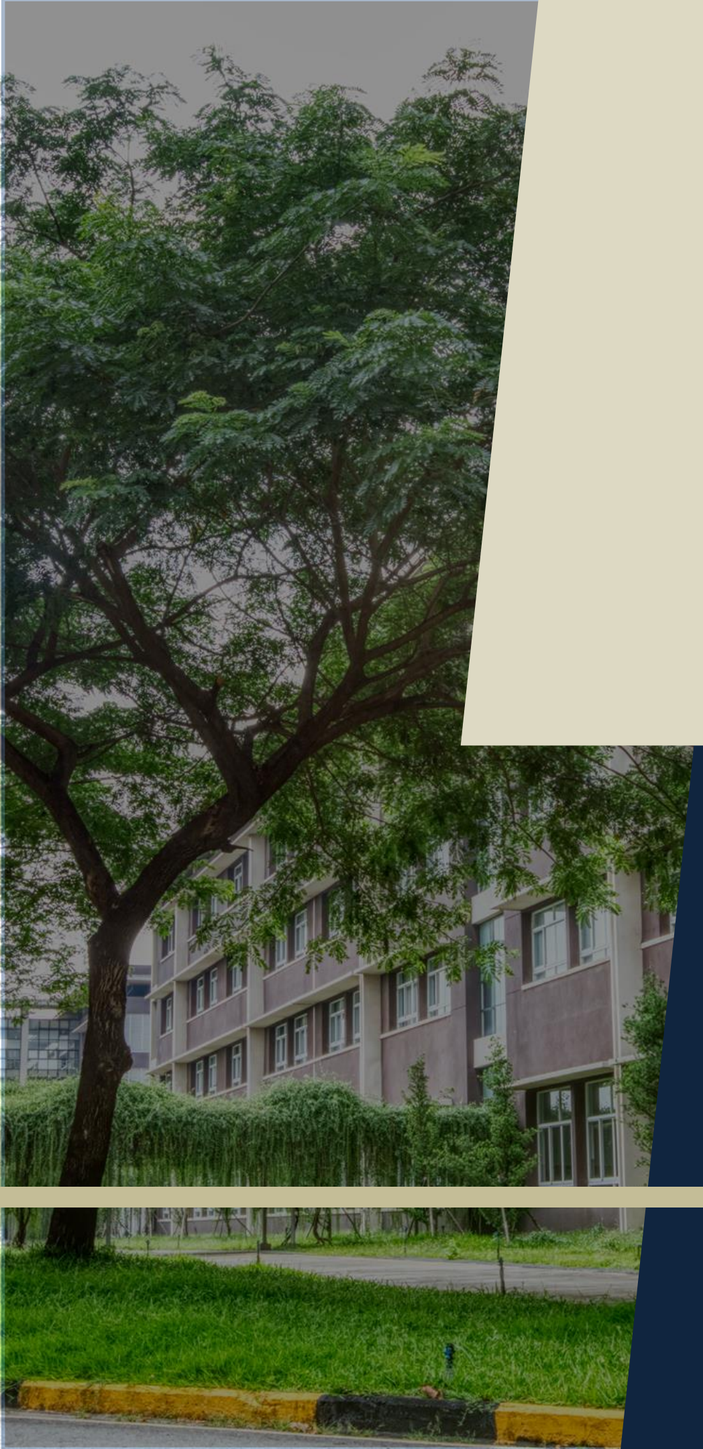
Model – Dependency injection in the controller



ASP.NET Core MVC

Model – Show in view





EIU
TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG
EASTERN
INTERNATIONAL
UNIVERSITY

Start your future at EIU

Thank You