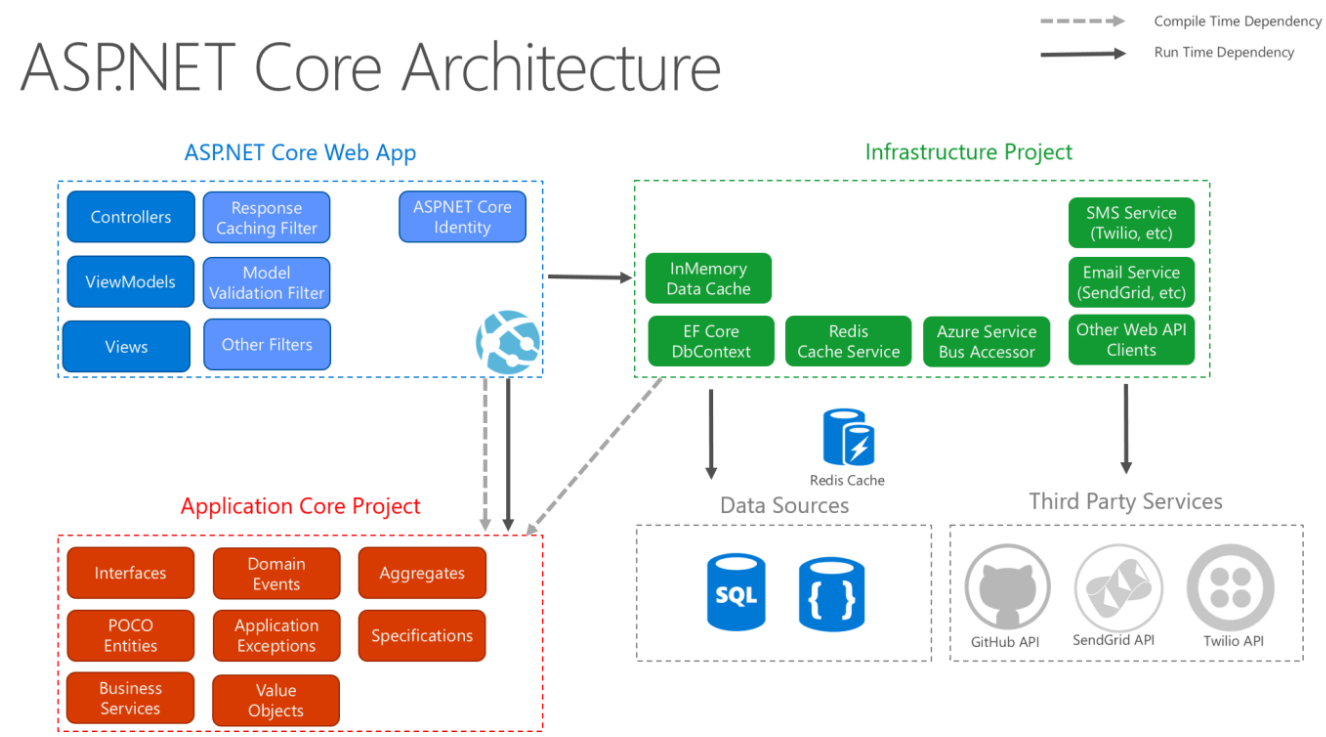
Clean Architecture .NET Core (Phần 2: Triển khai)

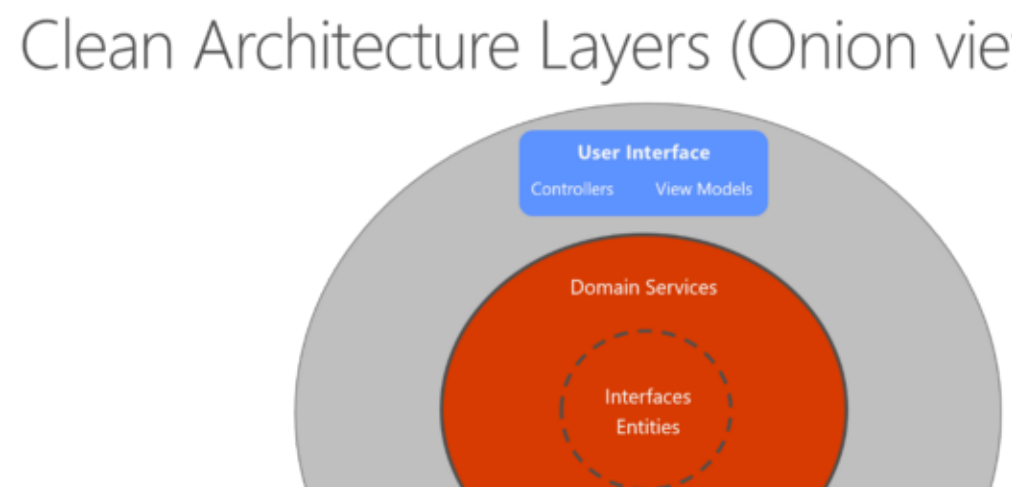
Nishan Chathuranga Wickramarathna Ngày 20 tháng 3 năm 2020 · 11 phút đọc

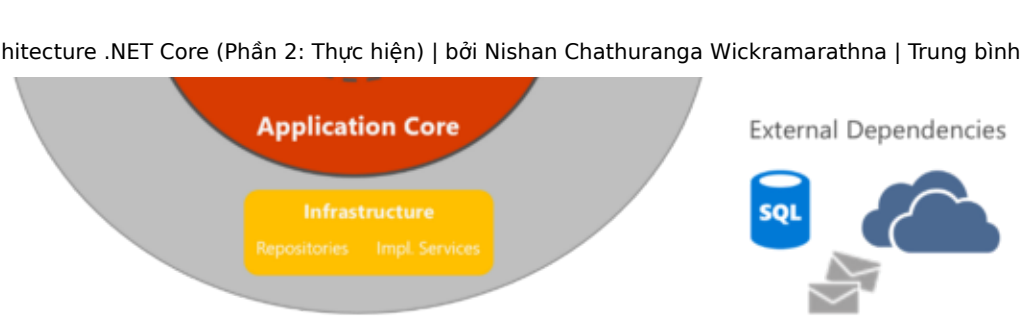


Xem chi tiết hơn về kiến trúc của ứng dụng ASP.NET Core khi được xây dựng theo các khuyến nghị kiến trúc sạch. Nguồn - Kiến trúc ứng dụng web phổ biến

Như bài viết trước, tôi đã giới thiệu cho bạn những cách thực hành cơ bản của Clean Architecture. Bây giờ chúng ta sẽ xây dựng một ứng dụng bằng ASP.NET Core 3, bắt đầu với cấu trúc thư mục. Tóm tắt nhanh trước khi tiếp tục.

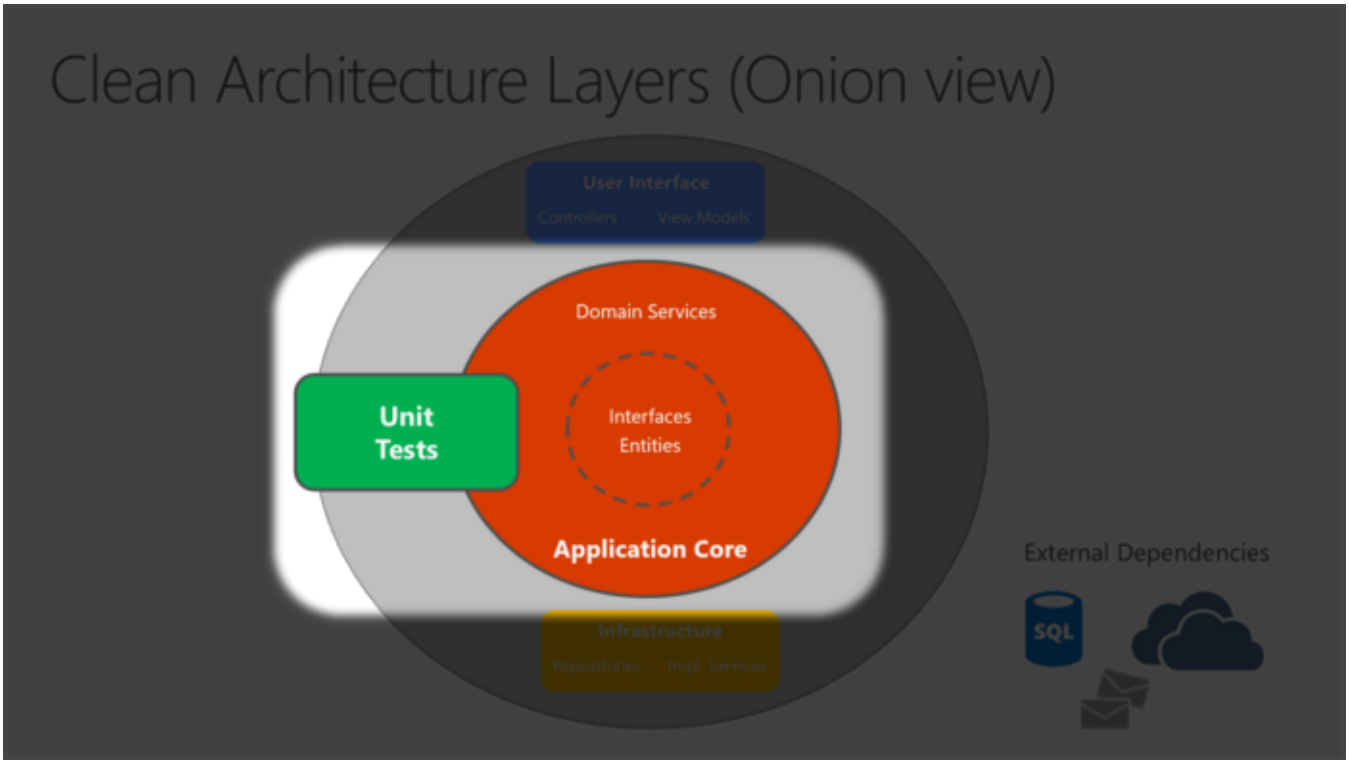
tóm tắt lại





Trong biểu đồ này, các phần phụ thuộc chảy về phía vòng tròn trong cùng. Lõi ứng dụng lấy tên từ vị trí của nó ở lõi của sơ đồ này. Và bạn có thể thấy trên sơ đồ rằng Lõi ứng dụng không có phụ thuộc vào các lớp ứng dụng khác. Các thực thể và giao diện của ứng dụng nằm ở trung tâm. Ngay bên ngoài, nhưng vẫn nằm trong Lõi ứng dụng, là các dịch vụ miền, thường triển khai các giao diện được xác định trong vòng tròn bên trong. Bên ngoài Lõi ứng dụng, cả giao diện người dùng và các lớp Cơ sở hạ tầng phụ thuộc vào Lõi ứng dụng, nhưng không phụ thuộc vào nhau (nhất thiết).

Vì Lõi ứng dụng không phụ thuộc vào Cơ sở hạ tầng nên rất dễ dàng để viết các bài kiểm tra đơn vị tự động cho lớp này.



Vì lớp giao diện người dùng không có bất kỳ sự phụ thuộc trực tiếp nào vào các loại được xác định trong dự án Cơ sở hạ tầng, nên cũng rất dễ dàng hoán đổi các triển khai, để tạo thuận lợi cho thử nghiệm hoặc để đáp ứng các yêu cầu ứng dụng thay đổi. Việc sử dụng và hỗ trợ chèn phụ thuộc được tích hợp trong ASP.NET Core làm cho kiến trúc này trở thành cách thích hợp nhất để cấu trúc các ứng dụng nguyên khối không tầm thường.

Điều kiện tiên quyết

1. Visual Studio
2. .NET Core SDK (đảm bảo tìm SDK hỗ trợ phiên bản Visual Studio bạn đang sử dụng, tôi đang sử dụng Visual Studio 2019 và .NET Core 3.1)
3. Hiểu biết cơ bản về .NET Core Web Applications, MVC, C #, SQL Server, Migrations, Identity, Entity Framework và Visual Studio

Nếu bạn đang sử dụng Visual Studio 2017 trở xuống, .NET Core 2.1 sẽ là phiên bản được hỗ trợ. Nhận đúng phiên bản

> ***Bài viết này nói về cách thiết lập cấu trúc dự án cho Kiến trúc sạch, không phải về .NET Core***

Tôi khuyên bạn nên sao chép dự án này và đọc bài viết từng bước bằng cách kiểm tra dự án, để tránh nhầm lẫn.

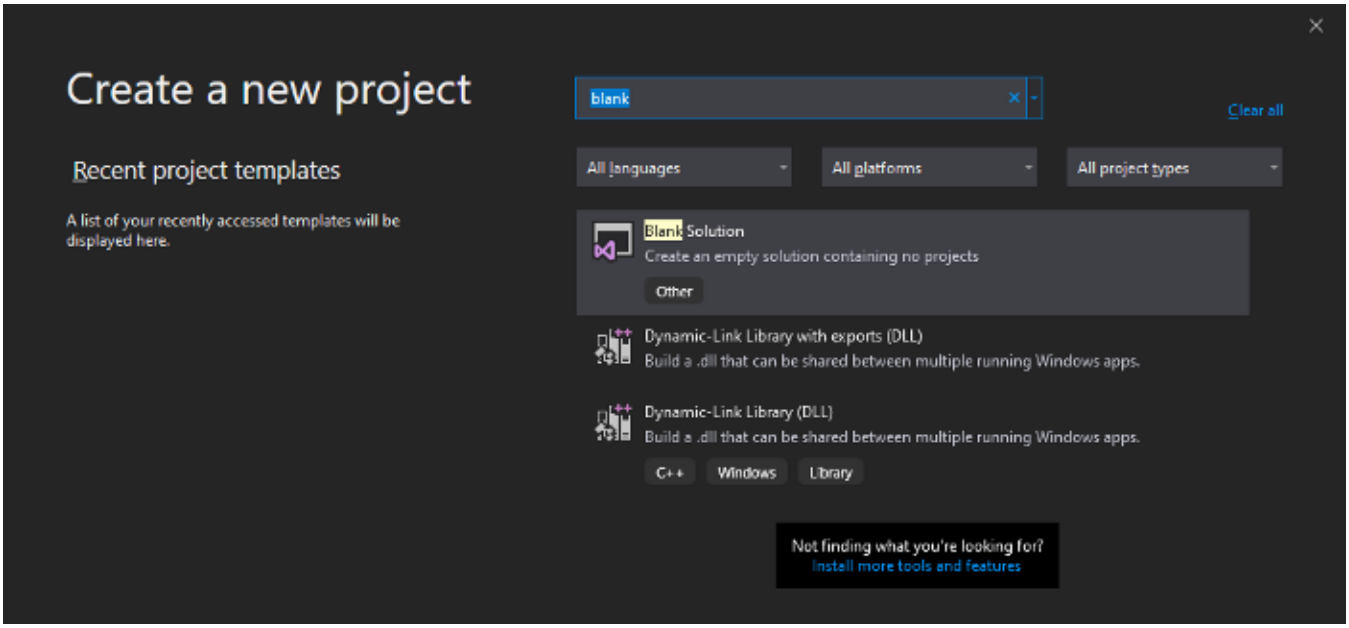
nishanc / CleanArchitectureDemo

Dự án demo được xây dựng trên .NET Core Clean Architecture - nishanc / CleanArchitectureDemo

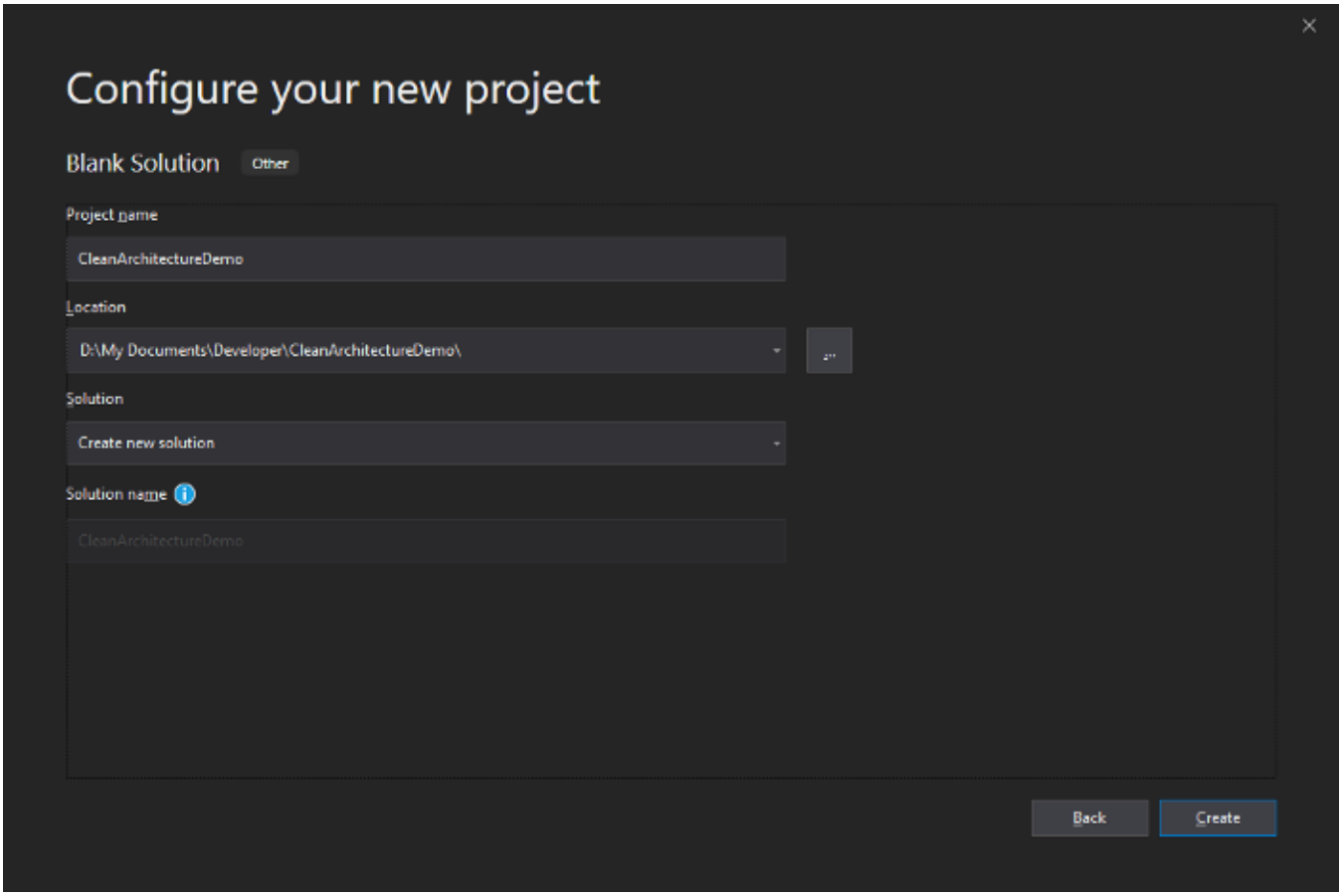
github.com

Cấu trúc thư mục

Tạo một giải pháp trống.

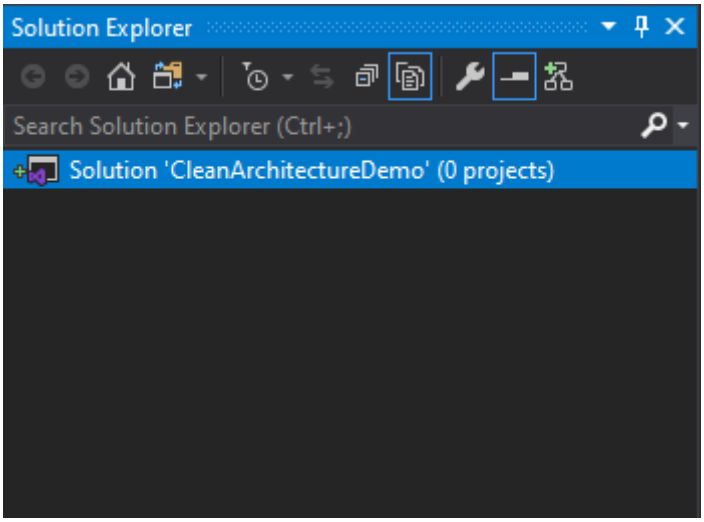


Tìm kiếm dự án giải pháp trống trong Tệp -> Mới -> Dự án. Chọn 'Giải pháp trống' và nhấp vào 'Tiếp theo'.



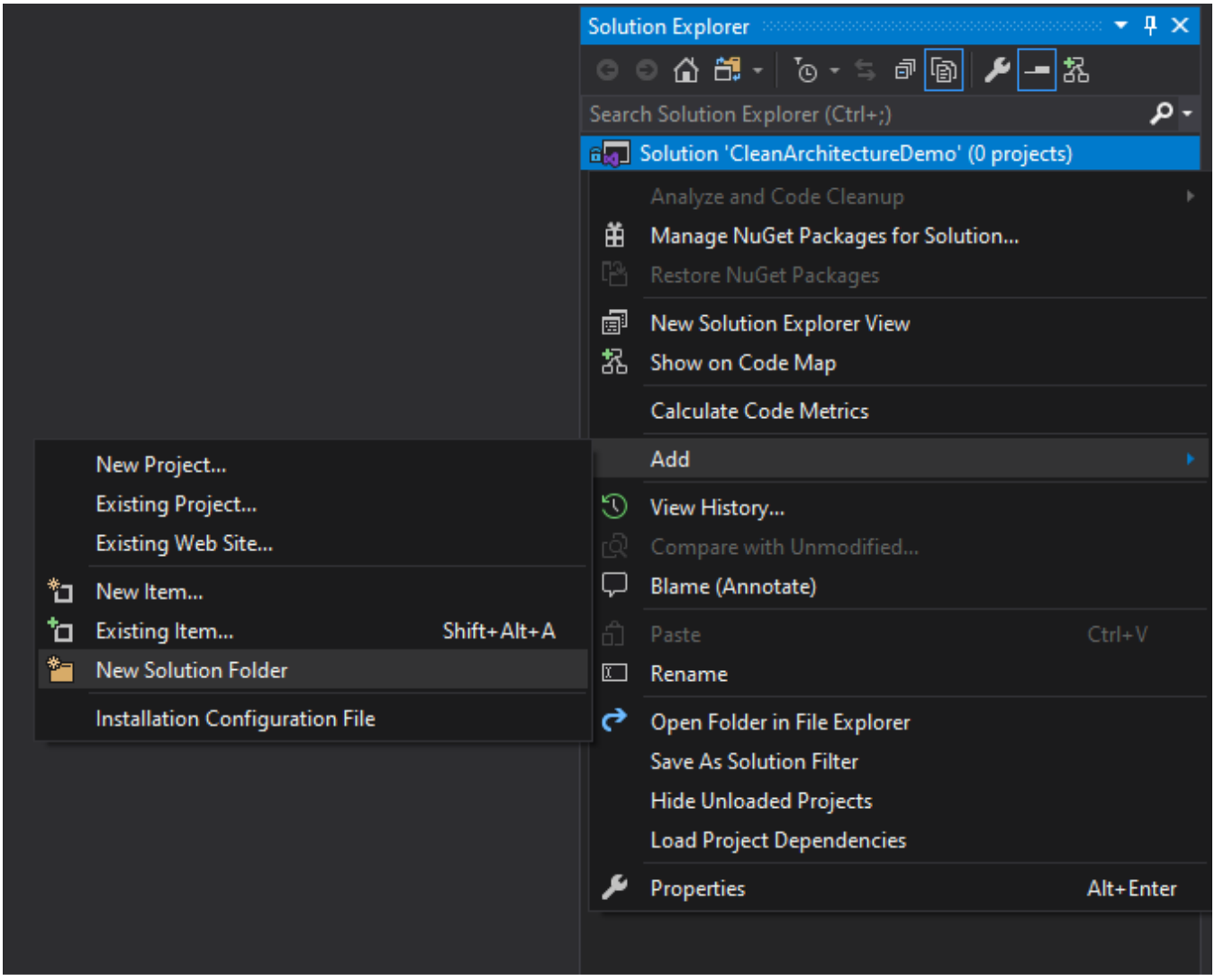
Đặt tên cho dự án của bạn, duyệt qua vị trí và nhấp vào 'Tạo'.

Bây giờ trình khám phá giải pháp của bạn sẽ trông như thế này.

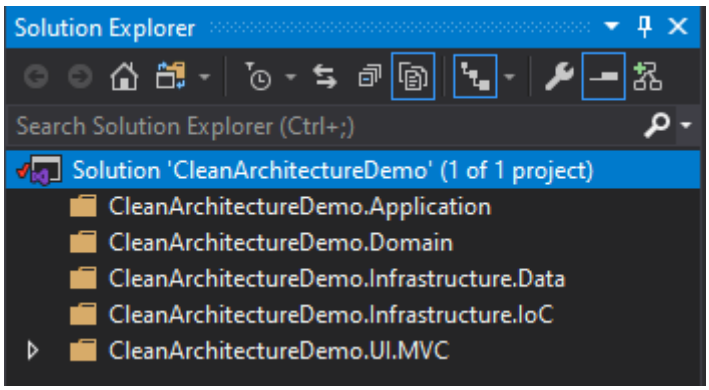


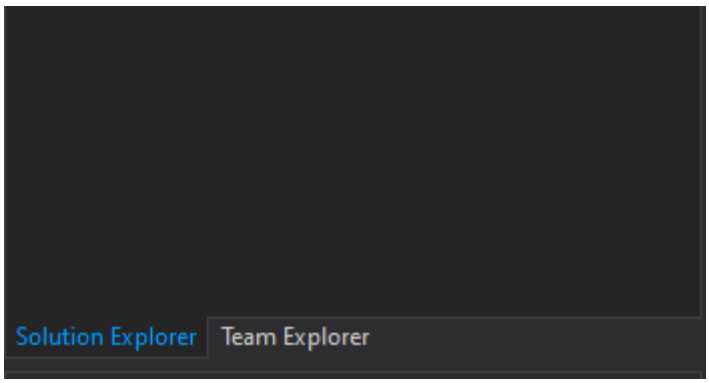
Bây giờ, hãy tạo các thư mục mà chúng ta sẽ sử dụng để lưu trữ các dự án riêng lẻ.

Tạo các thư mục sau bên trong giải pháp của bạn, đây chỉ là một gợi ý.



Nhấp chuột phải vào giải pháp -> Thêm -> Thư mục Giải pháp Mới



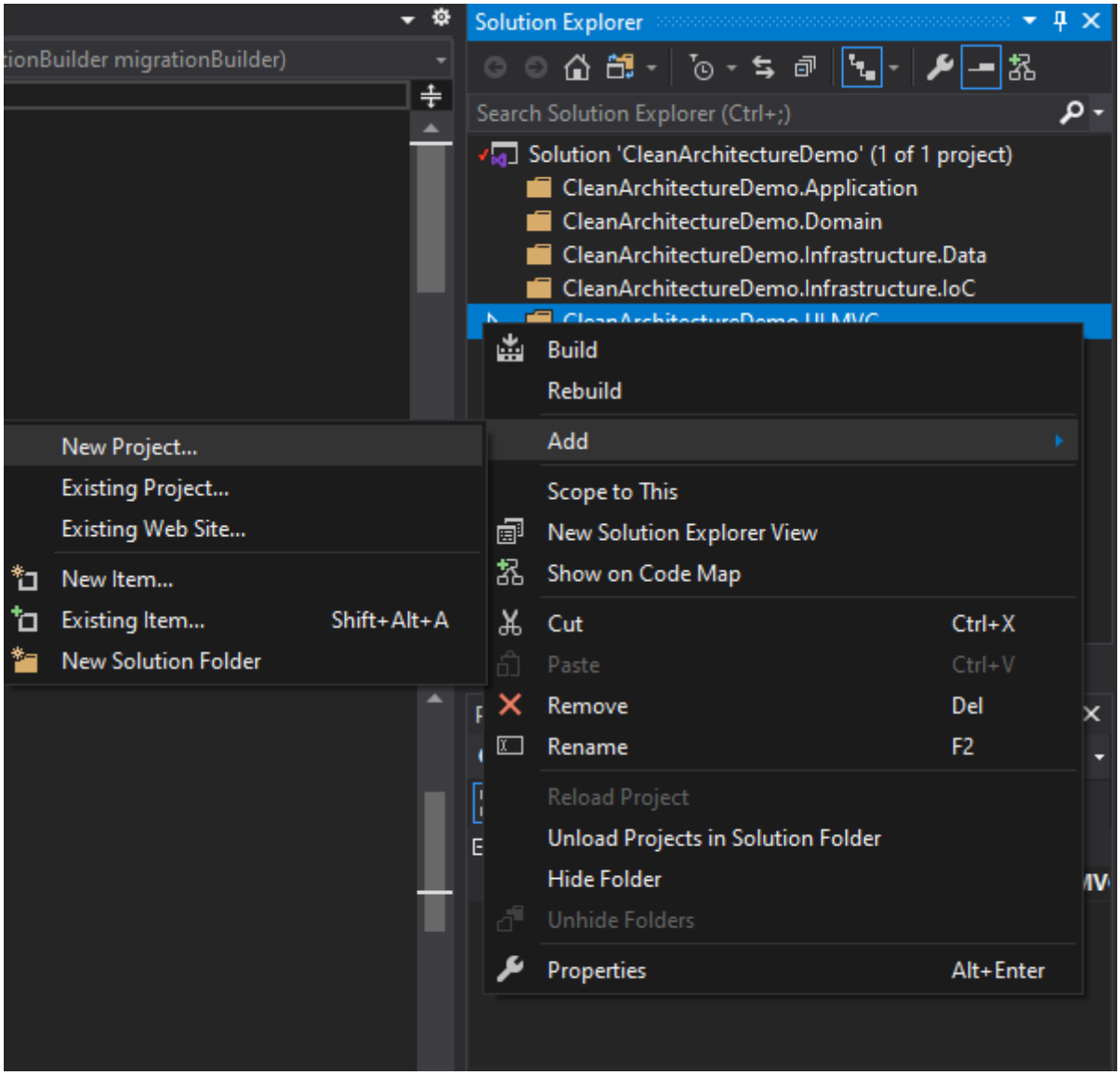


Tạo thư mục như đã thấy ở đây.

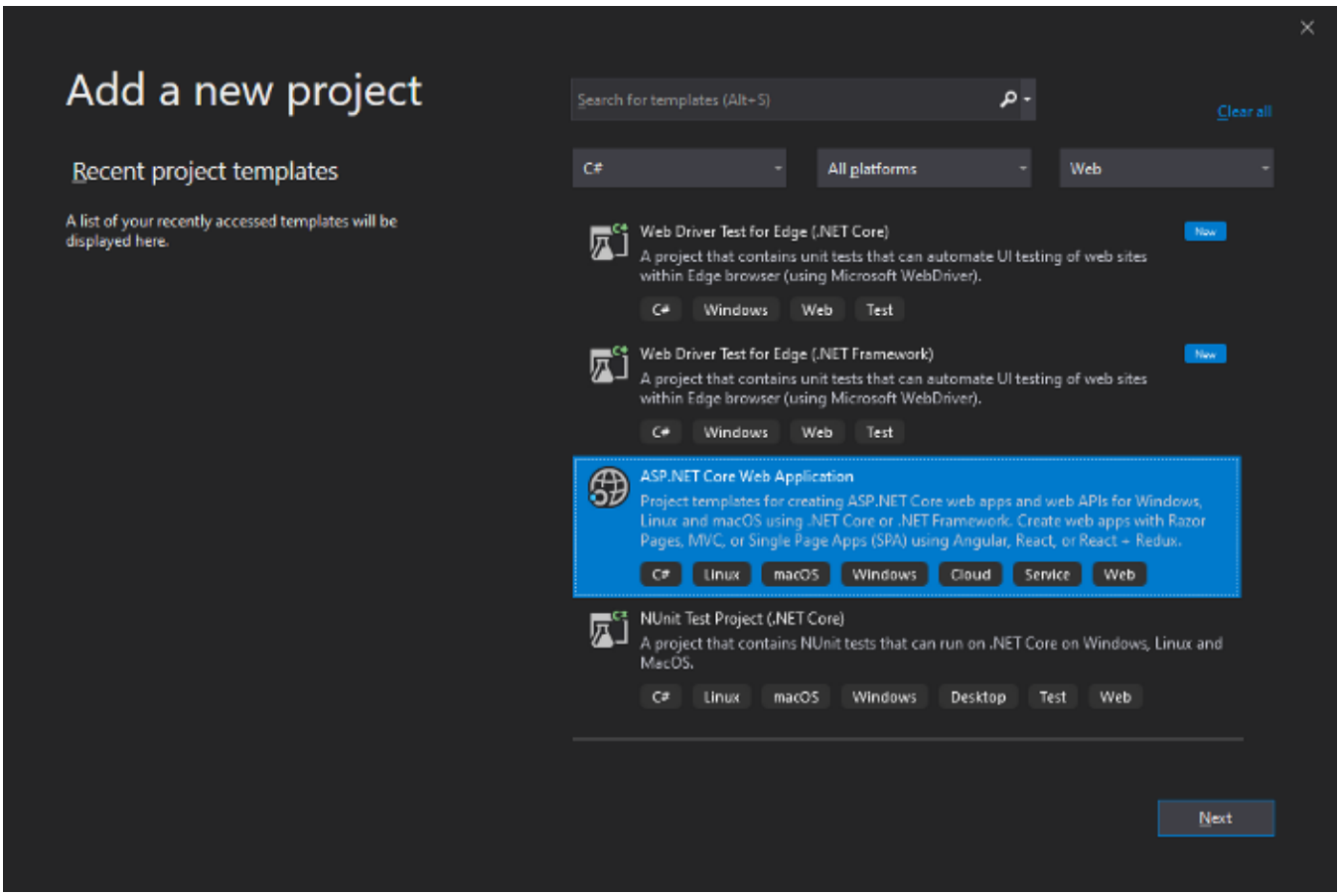
Ứng dụng sẽ quan tâm đến giao diện, dịch vụ, quy tắc kinh doanh của chúng tôi. **Miền**bao gồm các Thực thể, **Dữ liệu** biết về cách truy cập dữ liệu của chúng ta, **IoC** (Inversion of Control) sẽ giúp chúng ta đưa vào phụ thuộc.

Giao diện người dùng

Hãy tạo ứng dụng web MVC trong thư mục UI.MVC.

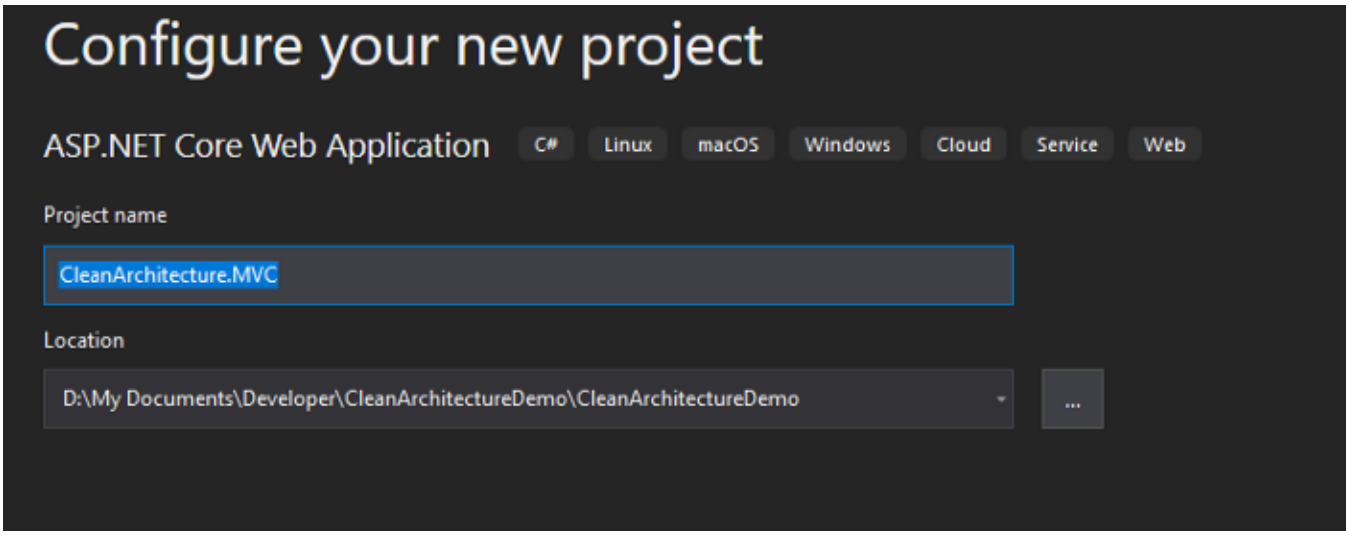


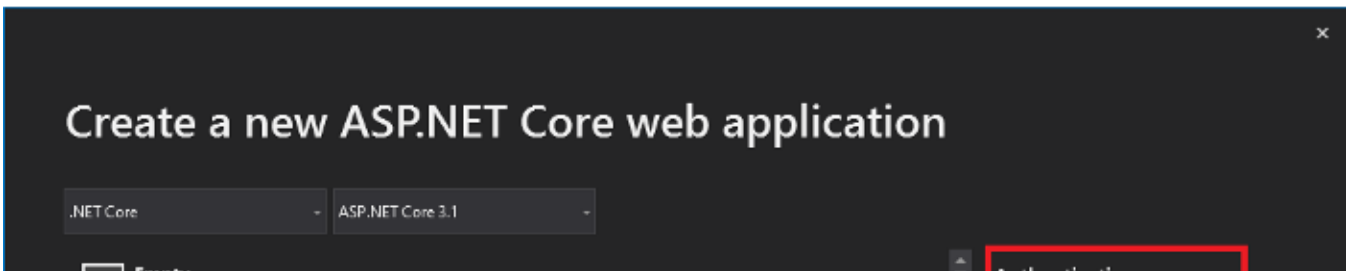
Nhấp chuột phải vào “CleanArchitecture.UI.MVC” -> Thêm -> Dự án mới

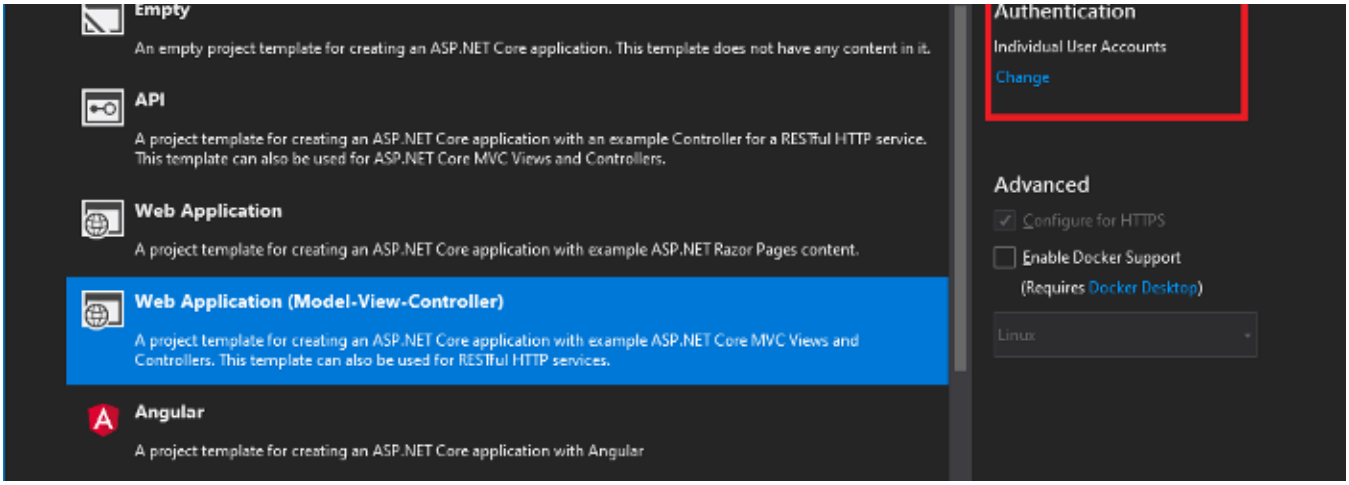


Chọn Ứng dụng Web ASP.NET Core . (Làm theo câu hỏi stackoverflow này nếu loại dự án này không hiển thị)

Theo cùng một quy tắc đặt tên mà chúng tôi đã làm theo, đặt tên dự án là 'CleanArchitecture.MVC' và nhấn 'Tạo'.







Chọn Ứng dụng Web (Model-View-Controller) và thay đổi Xác thực thành ' Tài khoản Người dùng Cá nhân '

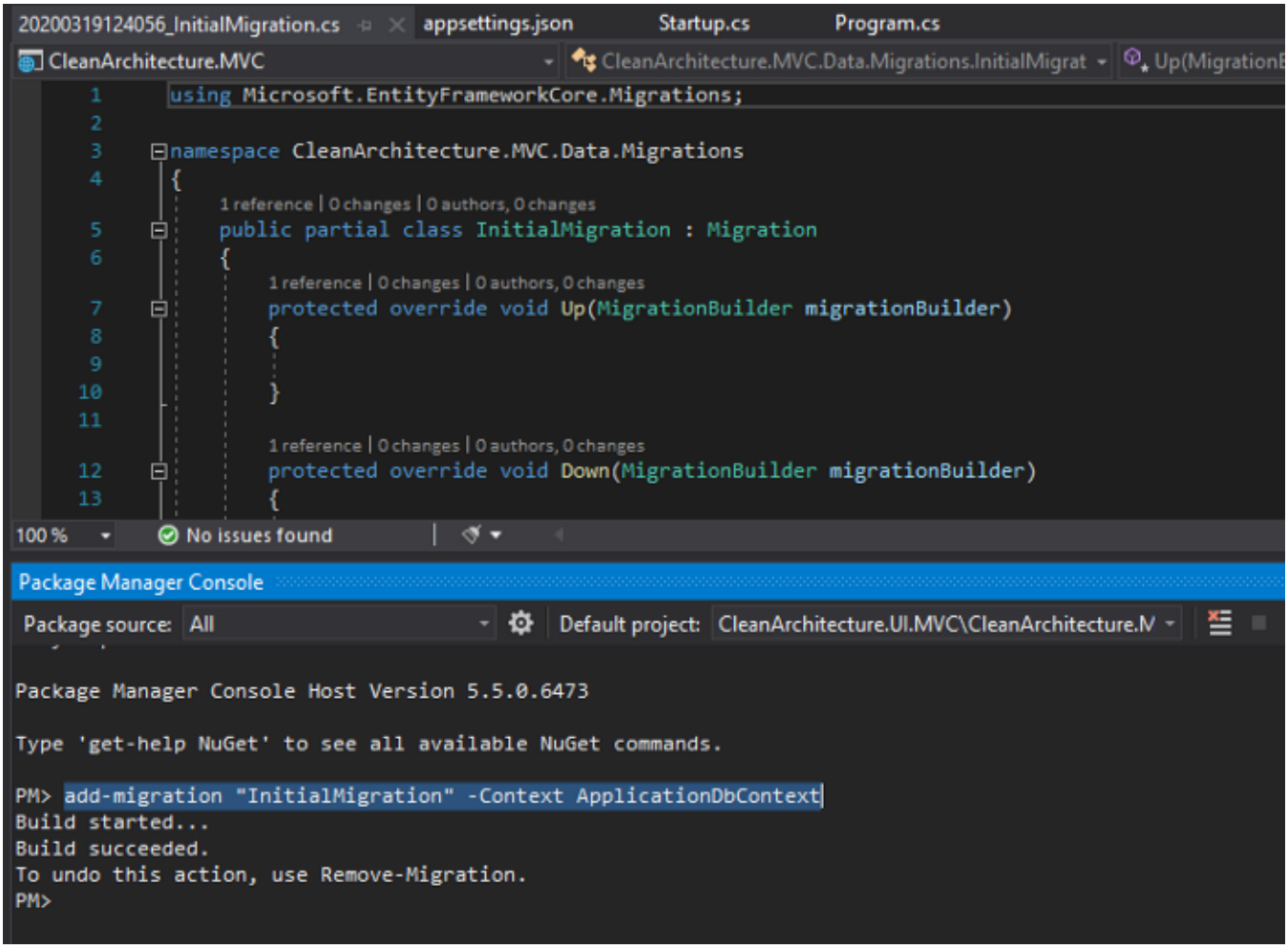
Thao tác này sẽ tạo một dự án mới và tạo đoạn mã cho một số đoạn mã. Sau đó, chúng tôi sẽ chuyển các mô hình sang các Thực thể sau, vì chúng tôi đang sử dụng cách tiếp cận Kiến trúc sạch. Sau khi hoàn tất, hãy nhanh chóng thay đổi chuỗi kết nối để trỏ đến cơ sở dữ liệu SQL Server. (Nếu bạn chưa cài đặt SQL Server, tôi khuyên bạn nên tải xuống **phiên bản dành cho nhà phát triển** từ đây và SQL Server Management Studio từ đây .)

Vào `appsettings.json` và thay đổi chuỗi kết nối để trỏ máy chủ của bạn, tôi sẽ tạo một cơ sở dữ liệu có tên là **Demo** .

Hãy cập nhật cơ sở dữ liệu với các lớp mô hình Identity bằng cách sử dụng di chuyển. Mở Bảng điều khiển Trình quản lý Gói (Công cụ -> Trình quản lý Gói NuGet -> Bảng điều khiển Trình quản lý Pakage.) Và nhập,

```
add -igration "InitialMigration" -Context ApplicationDbContext
```

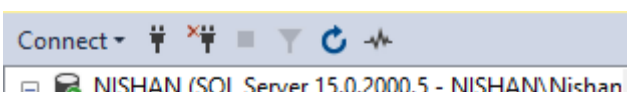
Lưu ý rằng chúng tôi đã chỉ định tên lớp DbContext ở đây, vì **chúng tôi sẽ** có một DbContext khác trong tương lai cho các thực thể khác. Lệnh này sẽ tạo một chuyển đổi có tên là “InitialMigration”, nó trông có vẻ trống nhưng nó sẽ tạo ra các bảng Identity mà chúng ta muốn.

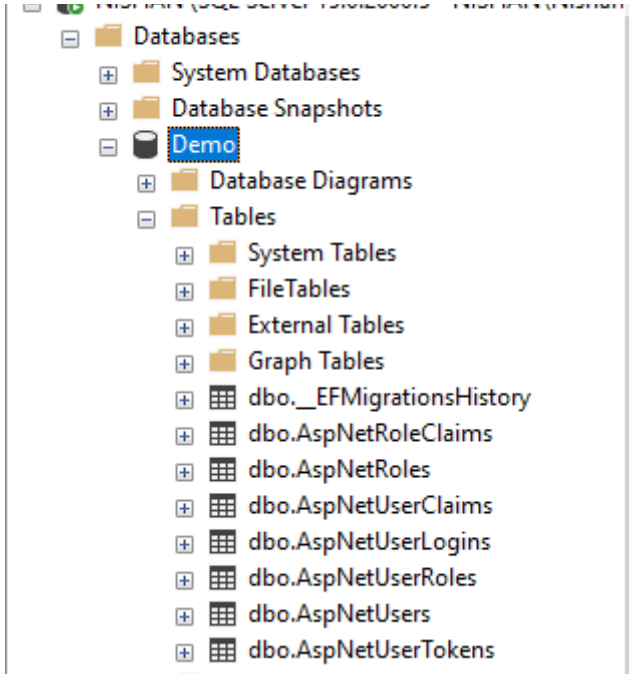


Bạn cũng có thể có một dự án khác cho Di chuyển danh tính nếu bạn muốn. Tôi sẽ trình bày cách thực hiện bằng cách chuyển các thực thể cốt lõi ra khỏi dự án MVC. Nhưng trước tiên, sau khi tạo quá trình di chuyển, hãy cập nhật cơ sở dữ liệu.

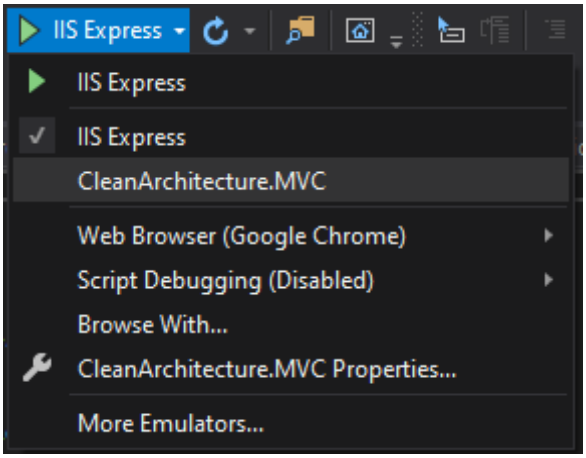
```
cập nhật cơ sở dữ liệu
```

Sau khi hoàn tất, bạn sẽ thấy cơ sở dữ liệu của mình có các bảng mới.





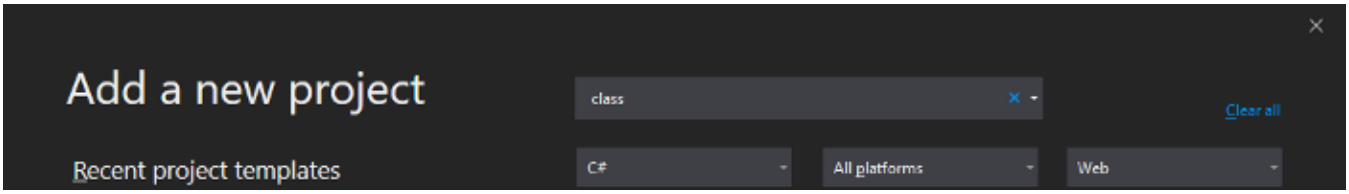
Tại thời điểm này, bạn có thể chạy dự án này bằng cách thay đổi cấu hình khởi chạy thành 'CleanArchitecture.MVC' từ thanh công cụ bị vu khống và đăng ký người dùng mới và xem điều đó có hoạt động hay không.

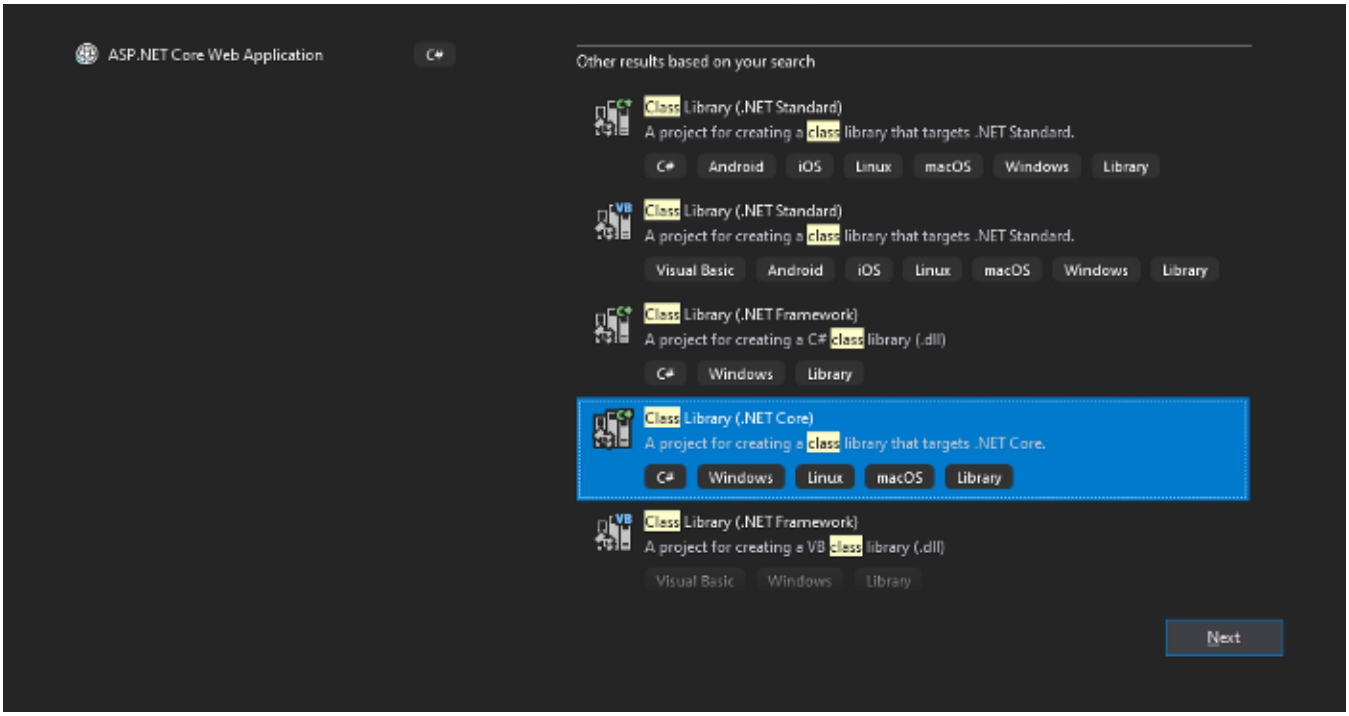


Tôi nên chỉ ra rằng bạn có thể đổi tên các thư mục giải pháp của mình để phản ánh ứng dụng của bạn, không cần phải theo cách đặt tên của tôi, nhưng tất nhiên là tuân theo kiến trúc sạch.

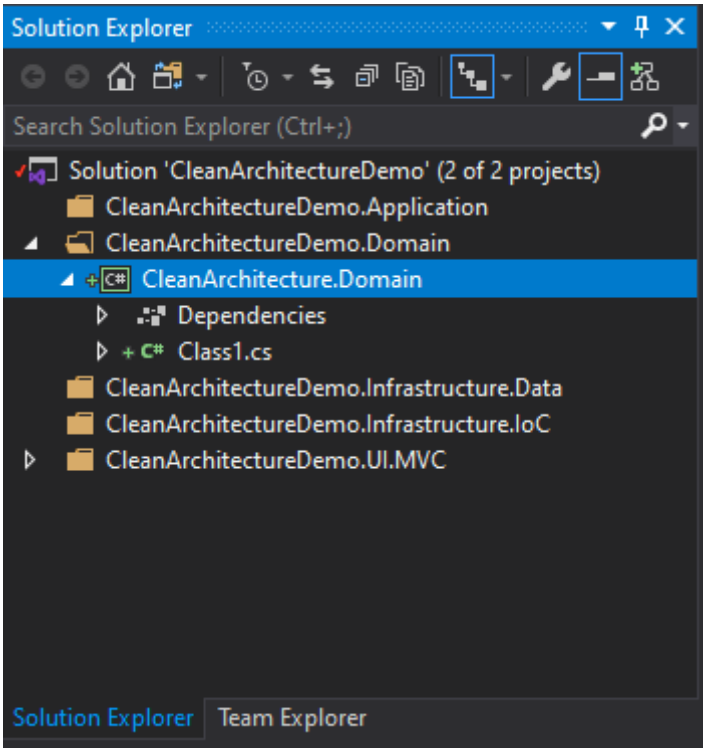
Lãnh địa

Hãy thêm một dự án Thư viện lớp vào lớp miền của chúng ta, lớp này sẽ chứa các thực thể cốt lõi.



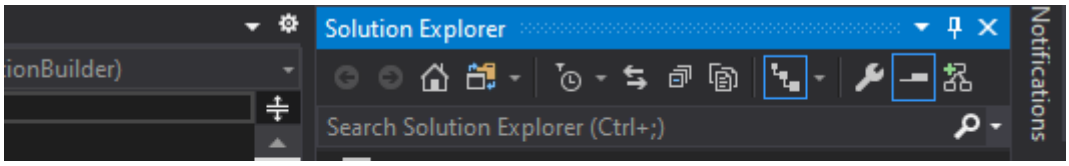


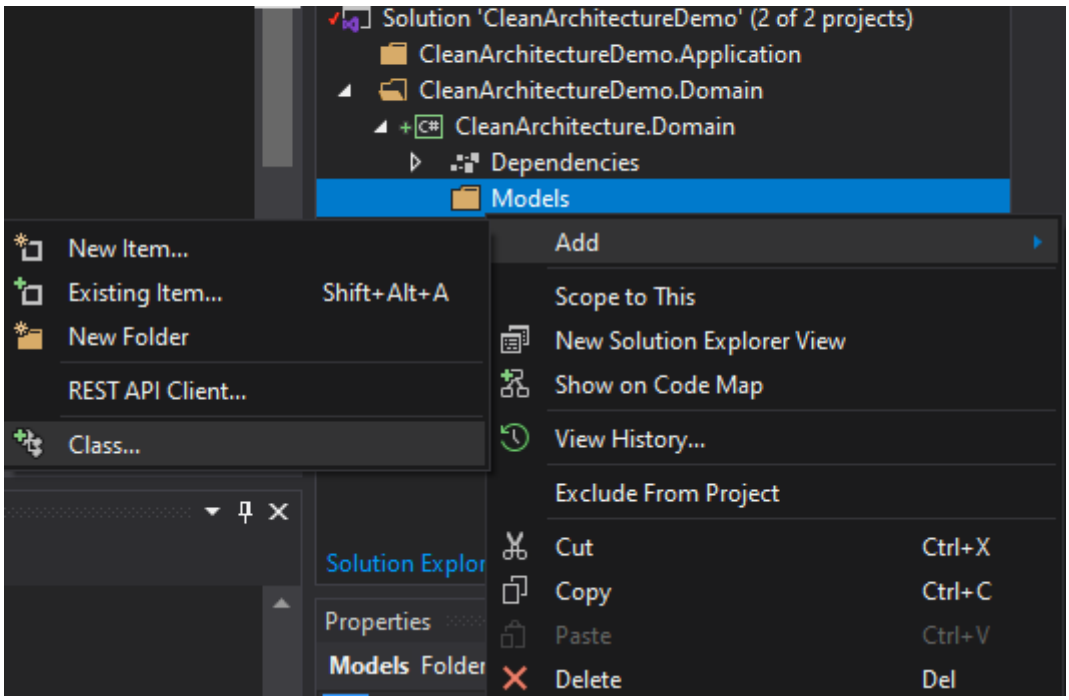
Chọn Thư viện Lớp (.NET Core)



Đặt tên nó là hậu tố Miền

Loại bỏ **Class1.cs** và thêm một thư mục mới **Mô hình** (hoặc Thực thể), vì lợi ích của bài viết này, hãy giả sử chúng tôi đang xây dựng một hệ thống thư viện, vì vậy chúng tôi sẽ thêm một mô hình **Sách** vào thư mục Mô hình.



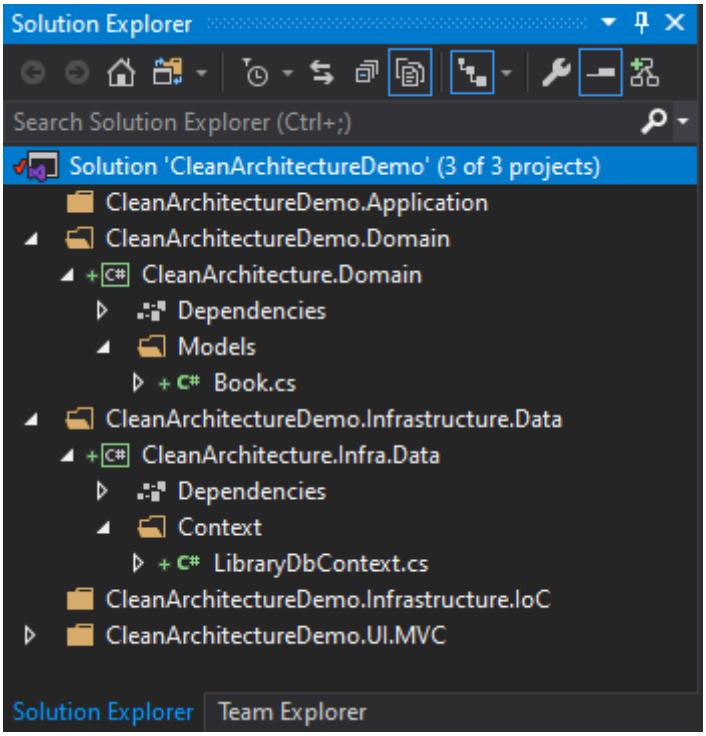


Nhấp chuột phải -> Thêm -> Lớp, đặt tên là Sách và nhấp vào Thêm .

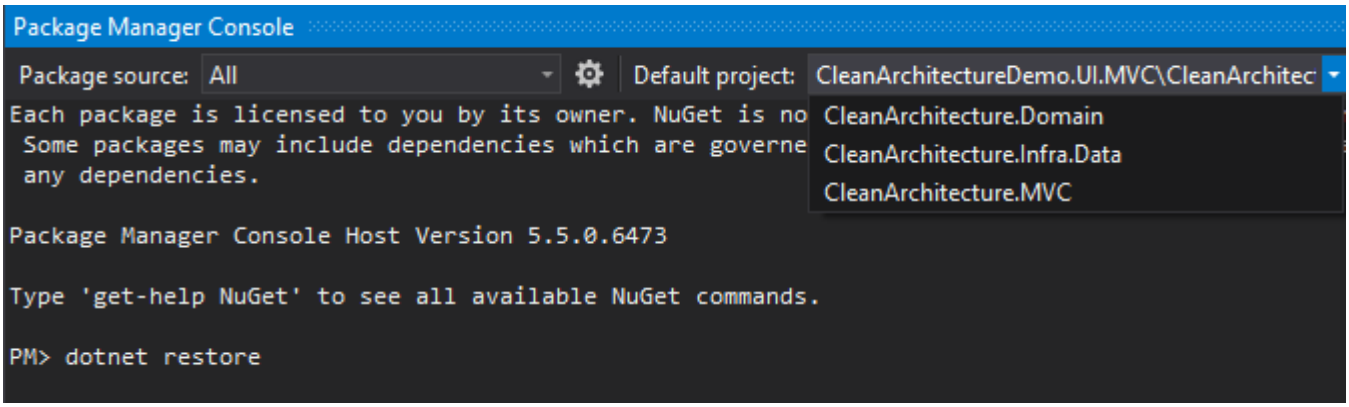
Thêm các thuộc tính bạn muốn vào thực thể có.

Lưu ý rằng tôi đã **công khai** lớp học **.**

Bây giờ chúng ta hãy tạo một bối cảnh cơ sở dữ liệu mới để chúng ta có thể cập nhật cơ sở dữ liệu với mô hình mới này. Chúng tôi sẽ làm điều đó trong lớp **Cơ sở hạ tầng** của chúng tôi . Trong **Infrastructure.Data,** thêm thư viện lớp mới có tên **CleanArchitecture.Infra.Data** và trong thư mục đó có tên là **Context** . Tiếp theo tạo một lớp có tên **LibraryDbContext.**



Hãy tiếp tục và thêm các gói chúng ta cần để cấu hình dự án này. Nhưng trước đó, trong Bảng điều khiển Trình quản lý Gói, hãy thực thi `dotnet restore` cho cả 3 dự án mà chúng tôi có cho đến nay. Sau đó thực hiện **Xây dựng -> Xây dựng lại Giải pháp**



Nhấp chuột phải vào **Dependencies** of **CleanArchitecture.Infra.Data** và chọn **Manage NuGet Packages** . Sau đó nhấp vào tab duyệt và cài đặt,

```
Microsoft.EntityFrameworkCore
Microsoft.EntityFrameworkCore.Design
Microsoft.EntityFrameworkCore.SqlServer
Microsoft.EntityFrameworkCore.Tools
```

> ***Đảm bảo rằng tất cả chúng đều là phiên bản ổn định (không phải bản xem trước) và ở cùng một phiên bản.***

Bây giờ chúng ta có thể kế thừa LibraryDbContext của mình từ DbContext do **Microsoft.EntityFrameworkCore** cung cấp . Và bạn có thể thấy rằng chúng tôi đang đề cập đến thực thể Sách được cung cấp bởi **CleanArchitecture.Domain.Models.** Thêm nội dung sau vào **LibraryDbContext**

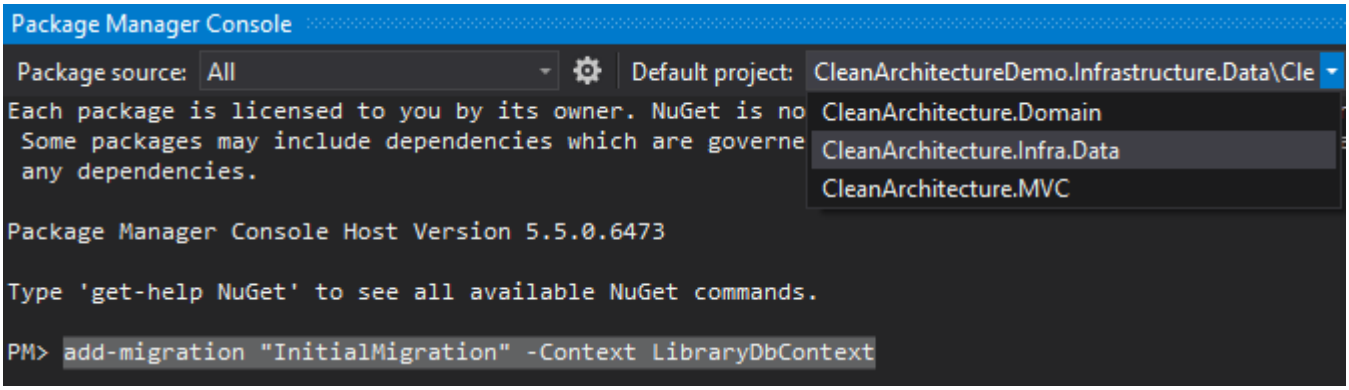
Hãy cấu hình DbContext mới này trong Startup.cs của dự án MVC. Thêm các dòng sau vào phương thức **ConfigureServices** .

Lưu ý đến chuỗi kết nối **LibraryConnection** , vì vậy chúng ta cần thêm nó vào appsettings.json. Chuỗi kết nối mới này trỏ đến một cơ sở dữ liệu khác, nơi chúng tôi sẽ lưu trữ các thực thể miền của mình.

```
"ConnectionStrings": {
    "DefaultConnection": "Server = .; Database = Demo;
Trusted_Connection = True; MultipleActiveResultSets = true",
    "LibraryConnection": "Server = .; Database = Library;
```

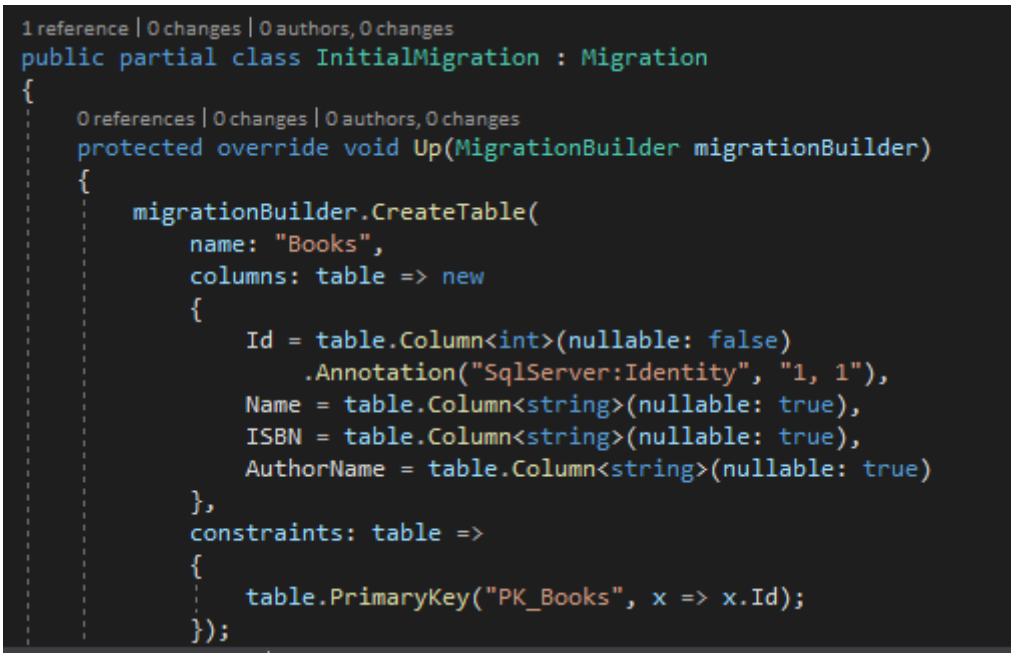
```
Trusted_Connection = True; MultipleActiveResultSets = true"
}
```

Sau khi hoàn tất, hãy thực thi `add-migration` với **LibraryDbContext**



Đảm bảo rằng bạn đã chọn CleanArchitecture.Infra.Data làm dự án mặc định.

Điều này sẽ tạo ra một cuộc di cư mới theo cơ sở hạ tầng.



```
1 reference | 0 changes | 0 authors, 0 changes
public partial class InitialMigration : Migration
{
    0 references | 0 changes | 0 authors, 0 changes
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.CreateTable(
            name: "Books",
            columns: table => new
            {
                Id = table.Column<int>(nullable: false)
                    .Annotation("SqlServer:Identity", "1, 1"),
                Name = table.Column<string>(nullable: true),
                ISBN = table.Column<string>(nullable: true),
                AuthorName = table.Column<string>(nullable: true)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_Books", x => x.Id);
            });
```

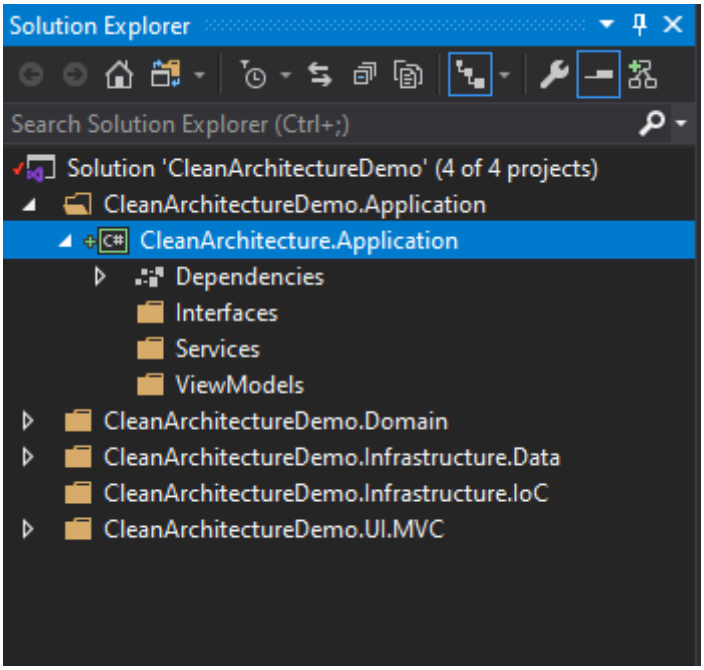
Cập nhật cơ sở dữ liệu bằng cách sử dụng,

```
cập nhật-cơ sở dữ liệu -Context LibraryDbContext
```

Lõi ứng dụng

Tất cả các dịch vụ, giao diện và ViewModels của chúng tôi sẽ xuất hiện tại đây. Giống như trong các bước trước, tạo Thư viện lớp mới (.NET Core) trong thư mục

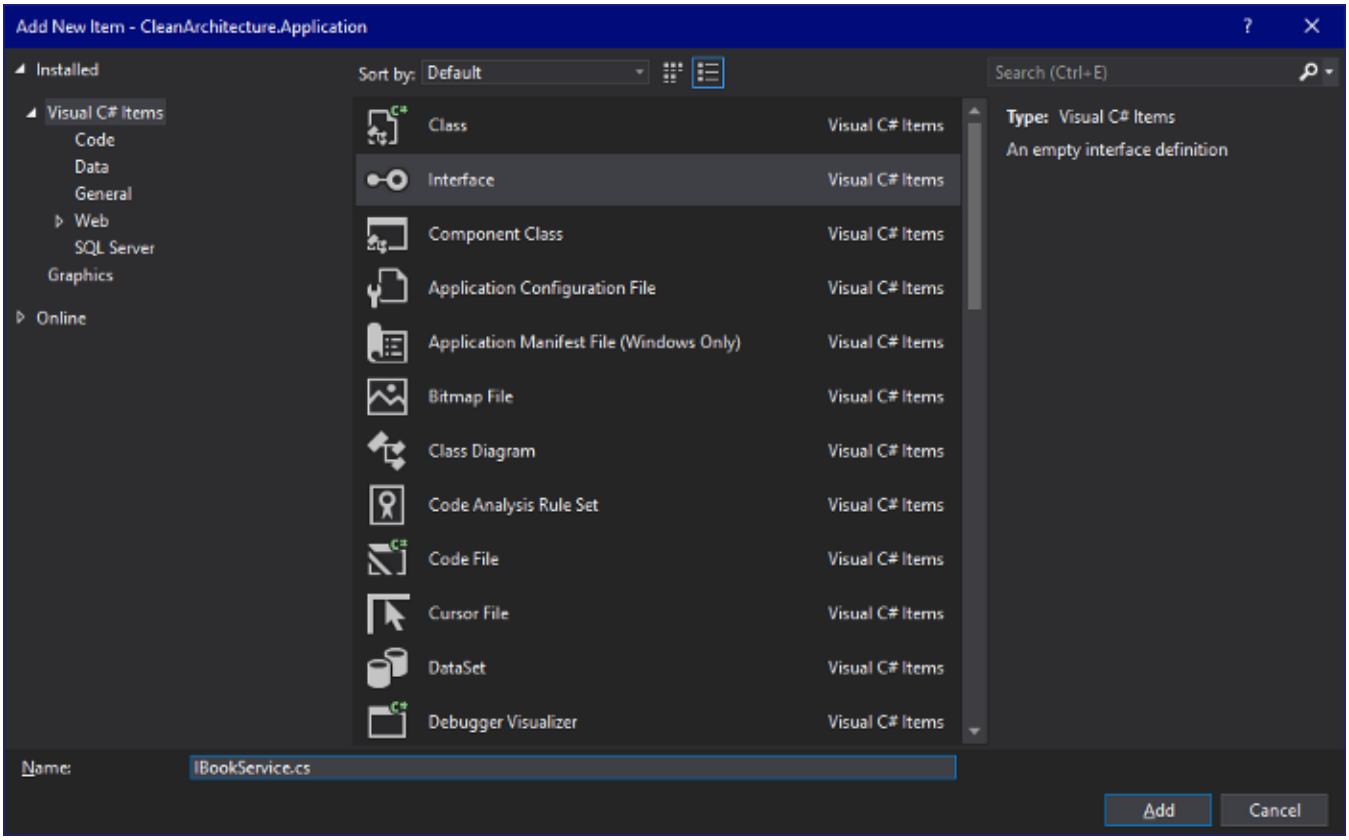
' **CleanArchitectureDemo.Application** ' có tên **CleanArchitecture.Application** và thêm các thư mục cho chúng như sau.



Hãy làm việc trên ViewModels ngay bây giờ, A `view model` đại diện cho dữ liệu mà bạn muốn hiển thị trên chế độ xem / trang của mình, cho dù nó được sử dụng cho văn bản tĩnh hay cho các giá trị đầu vào (như hộp văn bản và danh sách thả xuống) có thể được thêm vào cơ sở dữ liệu (hoặc chỉnh sửa). Nó là một cái gì đó khác với của bạn `domain model` . Nó là một mô hình cho khung nhìn. Nói cách khác, nó tạo ra một mặt nạ cho các mô hình miền.

Tạo một lớp mới trong thư mục ViewModels có tên **BookViewModel.** Tạm thời chúng ta sẽ lấy danh sách Sách từ cơ sở dữ liệu. Thêm mã sau vào **BookViewModel.cs** , thông báo rằng chúng tôi đang đưa **Sách** từ **CleanArchitecture.Domain.Models vào**

Tạo một giao diện mới để hoạt động như một hợp đồng với chức năng mà chúng tôi đang cố gắng triển khai. Tôi hy vọng bạn đã quen với các giao diện và lý do tại sao chúng tôi tạo giao diện trong OOP, vì những thứ đó nằm ngoài phạm vi của bài viết này, tôi sẽ không thảo luận ở đây. Trong thư mục **Interfaces,** tạo một **giao diện** mới ,



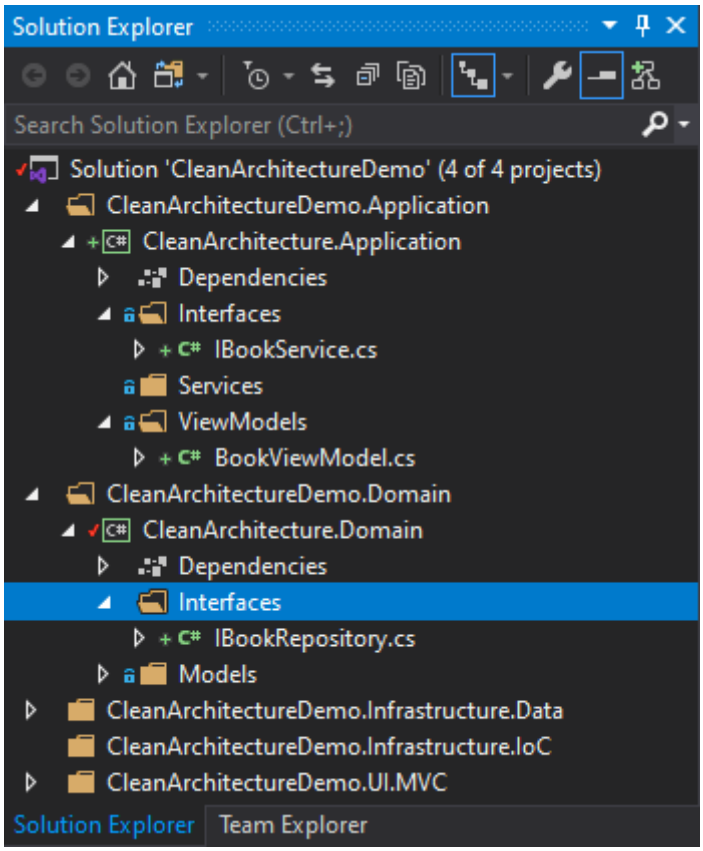
được đặt tên, **IBookService.cs**

Khi được triển khai, phương thức này sẽ trả về danh sách sách và nó chỉ biết về ViewModel, không biết về mô hình miền lõi Sách, vì vậy chúng tôi đang trừu tượng hóa thực thể cốt lõi bằng cách thực hiện điều này, thay vì đặt mọi thứ ở một nơi.

Trước khi chúng tôi viết triển khai cho IBookService, chúng tôi phải xác định một cách để lấy dữ liệu từ cơ sở dữ liệu, Để làm điều đó, những gì chúng tôi thường sử dụng trong .NET là ORM được gọi là Entity Framework, nhưng chúng tôi sẽ sử dụng **mẫu Kho lưu trữ** để tách doanh nghiệp. logic và các lớp truy cập dữ liệu trong ứng dụng của chúng tôi.

> **Mẫu thiết kế kho lưu trữ trong C # Dàn xếp giữa miền và các lớp ánh xạ dữ liệu bằng cách sử dụng giao diện giống như bộ sưu tập để truy cập các đối tượng miền. Nói cách khác, chúng ta có thể nói rằng Mẫu thiết kế kho lưu trữ hoạt động như một trung gian hoặc lớp trung gian giữa phần còn lại của ứng dụng và logic truy cập dữ liệu.**

Thêm một thư mục khác được gọi là Interfaces trong dự án **CleanArchitecture.Domain**. Thêm giao diện mới có tên **IBookRepository.cs**



Tại thời điểm này, dự án của chúng tôi sẽ trông như thế này.

Thêm phương thức sau, lưu ý lần này nó không phải là ViewModel, nó là thực thể miền, Book.

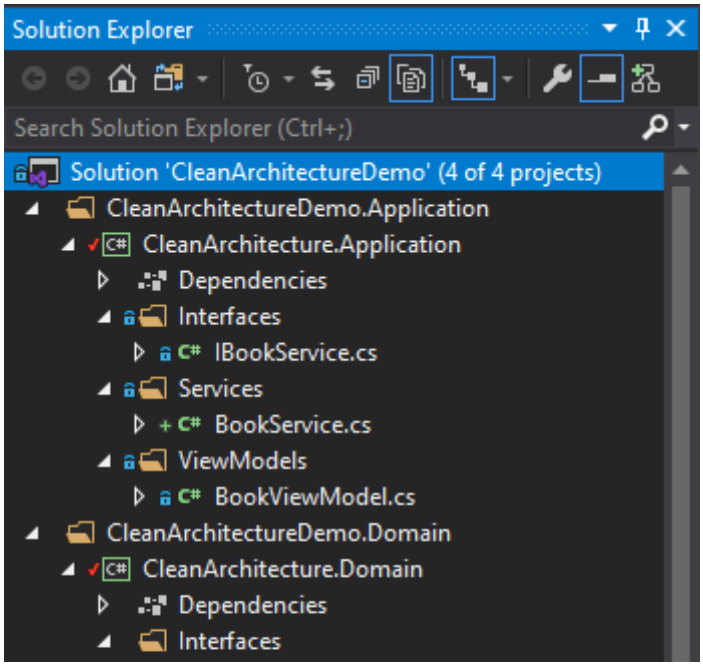
Được rồi, vậy tại thời điểm này, nếu dự án MVC hoặc **Lớp trình bày (** không có ý tưởng về thực thể miền **Sách)** nói, “này, tôi muốn có một danh sách sách!”, Nó cần phải nói chuyện với **BookService** (mà chúng tôi chưa được triển khai, sử dụng **IBookService**) và **BookService** cần lấy nó từ **BookRepository** (chúng tôi cũng chưa triển khai, sử dụng **IBookRepository**)

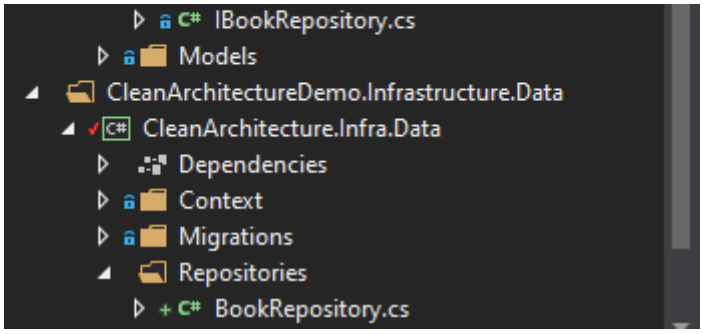
Vì vậy, hãy thực hiện chúng. Đầu tiên, **BookService.**

Trong dự án **CleanArchitecture.Application** , trong thư mục services, thêm một lớp mới, **BookService.cs** và kế thừa nó từ **IBookService.**

Bây giờ chúng ta cần phải đưa IBookRepository vào, hãy tiêm nó như cách bạn thường thực hiện việc tiêm phụ thuộc trong .NET

Tiếp theo **BookRepository** . Trong dự án **CleanArchitecture.Infra.Data** tạo một thư mục mới có tên là **Kho lưu trữ,** trong đó tạo lớp mới, **BookRepository** .





Giống như chúng ta đã triển khai **BookService,** chúng ta phải triển khai **BookRepository** từ **IBookRepository** , sau đó đưa ngữ cảnh cơ sở dữ liệu vào để chúng ta có thể nói chuyện với cơ sở dữ liệu.

Chúng tôi chưa triển khai các phương thức trong **BookService và BookRepository** , vì vậy hãy tiếp tục và triển khai các phương pháp đó.

Đầu tiên, hãy truy cập **BookRepository** và sử dụng ngữ cảnh được chèn, truy xuất sách trong cơ sở dữ liệu.

Tiếp theo, **BookService.**

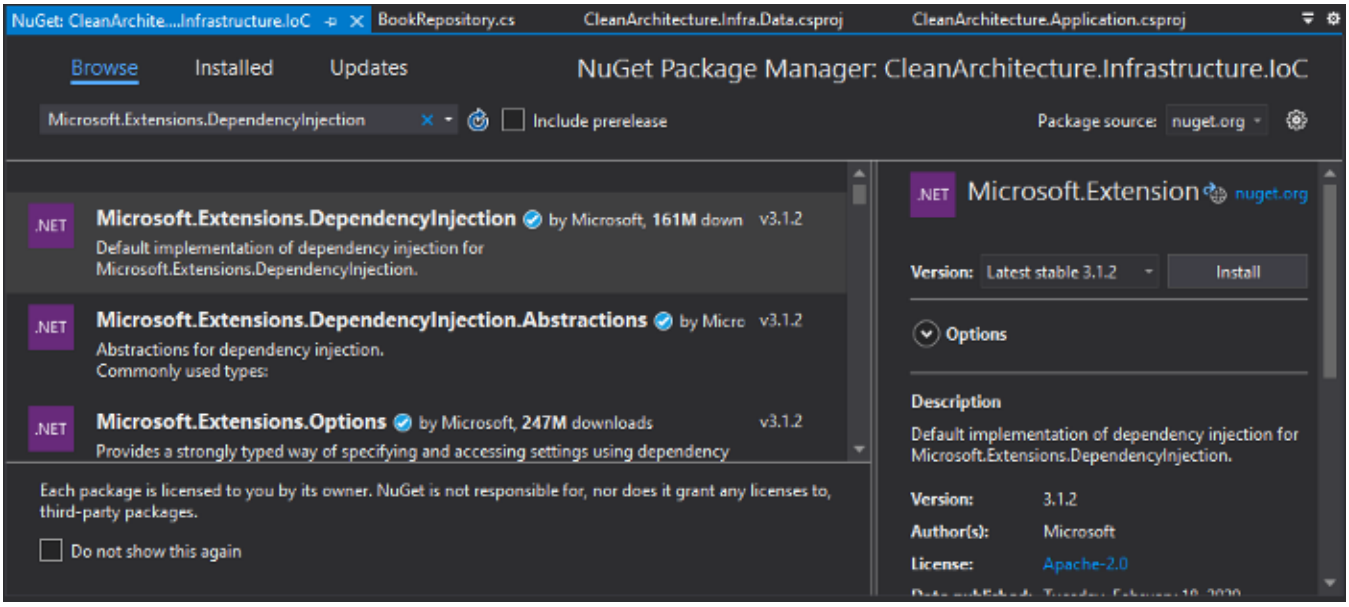
Tiếp theo, chúng ta cần xem xét việc thực hiện dự án IoC, dự án này sẽ giúp chúng ta chứa và tách các thành phần phụ thuộc.

Đảo ngược kiểm soát

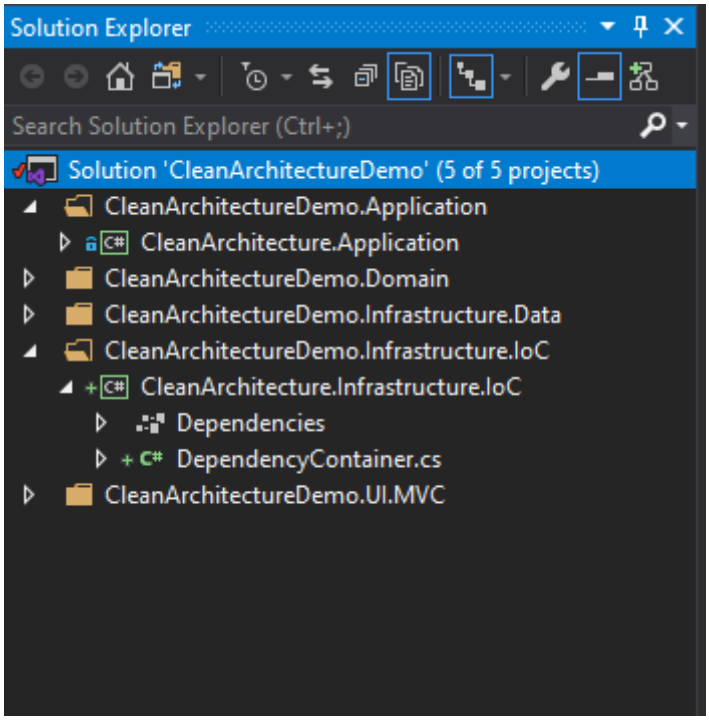
Vì vậy, trong **CleanArchitectureDemo.Infraosystem.IoC** , hãy tạo một Thư viện Lớp Core .NET mới giống như cách bạn đã làm trong các dự án trước đó có tên **CleanArchitecture.Infraosystem.IoC.** Thoát khỏi Class1.cs, nhấp chuột phải vào phụ thuộc -> Quản lý Gói NuGet -> chuyển đến tab duyệt và tìm kiếm,

```
Microsoft.Extensions.DependencyInjection
```

Đảm bảo rằng bạn cài đặt cùng một phiên bản với các phần phụ thuộc trước đó mà chúng tôi đã cài đặt và là các phiên bản ổn định.



Bây giờ chúng ta hãy tạo lớp Dependency Container, trong dự án **CleanArchitecture.Infraosystem.IoC** .



Lưu ý cách nó kết nối các giao diện của chúng tôi và việc triển khai chúng từ nhiều dự án thành một điểm tham chiếu duy nhất. Đó là mục đích của lớp IoC. Ngoài ra, hãy lưu ý **AddScoped,** nếu bạn muốn biết sự khác biệt, hãy đọc thêm tại đây.

Bây giờ chúng ta cần nói về vùng chứa dịch vụ mới này cho dự án MVC. Trụ sở để **Startup.cs,** sau khi **cấu hình** phương thức add phương pháp sau đây.

Bây giờ bên trong phương thức **ConfigureServices,** chúng ta cần gọi phương thức này.

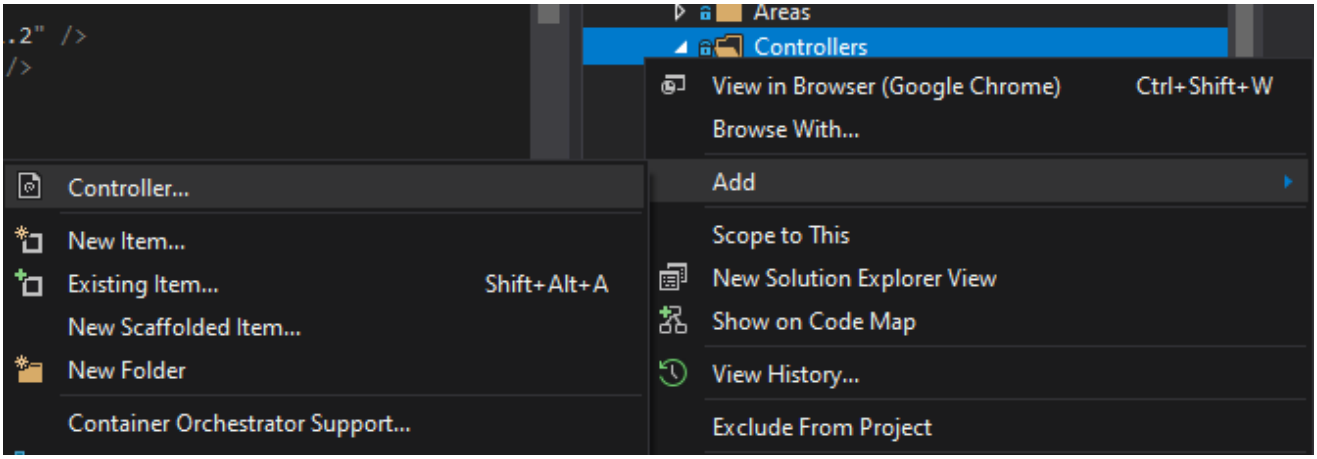
```
RegisterServices (dịch vụ);
```

Startup.cs đầy đủ trông như thế này .

Vì vậy, bây giờ tất cả các lớp của chúng tôi đã sẵn sàng. Vì vậy, tiếp theo chúng ta sẽ xem xét cách tạo **Bộ điều khiển** , sử dụng mọi thứ chúng ta đã thảo luận ở đây và triển khai giao diện người dùng.

Đầu tiên, hãy thêm một số dữ liệu vào cơ sở dữ liệu, vì chúng tôi chỉ triển khai phương thức **GetBooks** , chúng tôi cần thêm một số dữ liệu theo cách thủ công.

Khi bạn đã có dữ liệu trong cơ sở dữ liệu, hãy tạo một bộ điều khiển trong dự án **CleanArchitecture.MVC** , trong thư mục **Bộ điều khiển** .



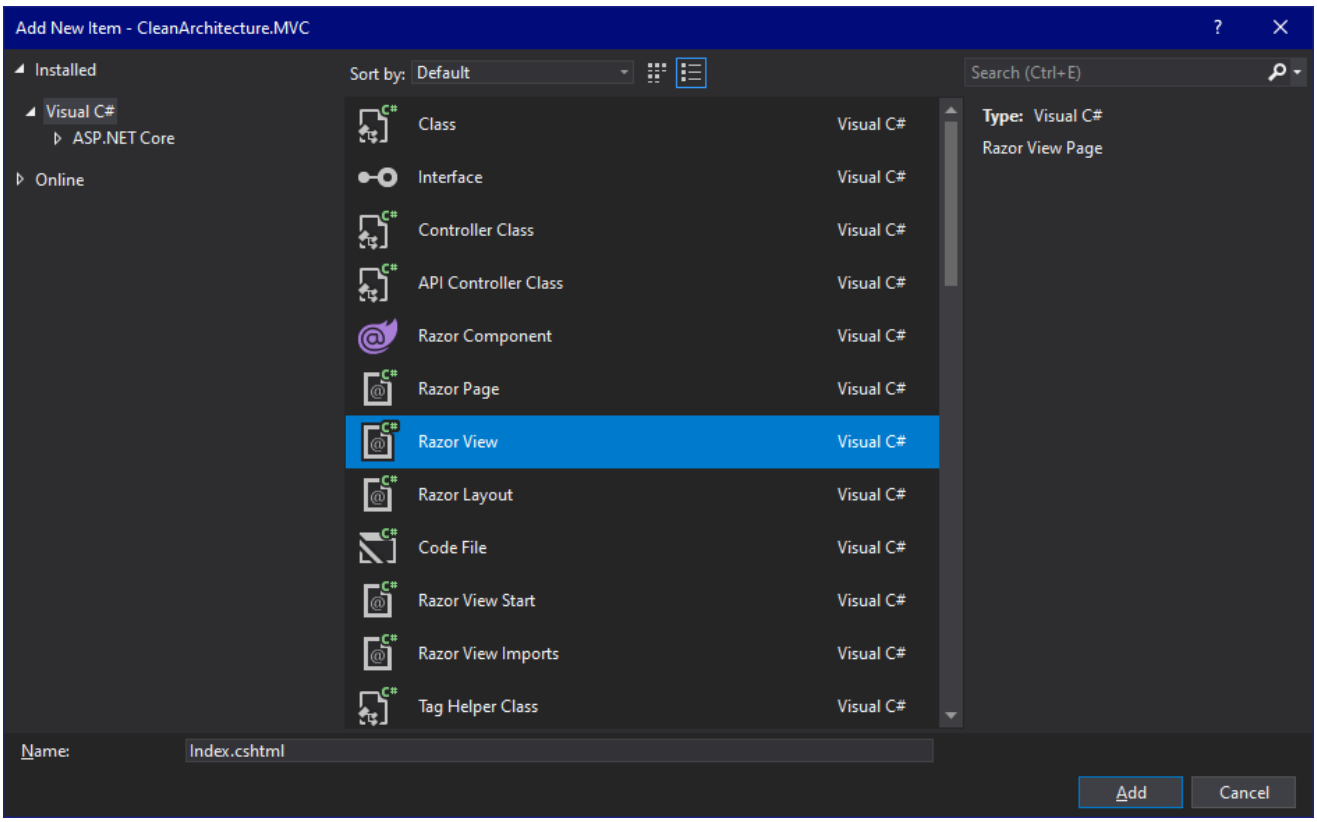
Nhấp chuột phải vào thư mục Controllers -> Add -> Controller, đặt tên là **BookController** theo quy ước MVC. Sau đó, chúng tôi sẽ đưa **BookService** vào bộ điều khiển này như sau.

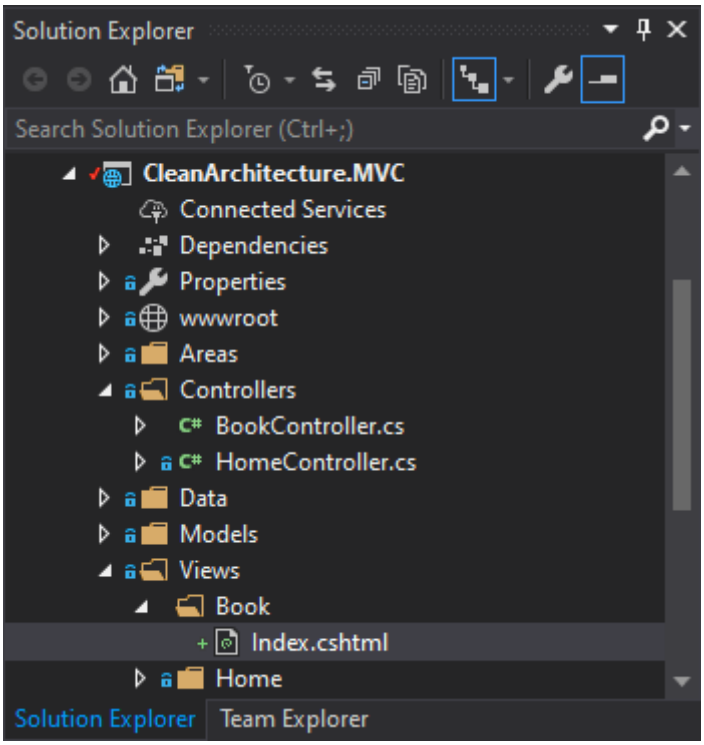
Lưu ý rằng chúng tôi đang đề cập đến IBookService trong lớp Ứng dụng.

Bây giờ chúng ta sẽ chuyển BookViewModel đến chế độ xem.

Xem nó trông **sạch sẽ** như thế nào . Đó là vẻ đẹp của kiến trúc sạch sẽ. Vì vậy, bây giờ chúng ta đã chuyển BookViewModel sang Chế độ xem của chúng ta, hãy tiếp tục và xác định chế độ xem đó.

Nhấp chuột phải vào thư mục **Views** , thêm thư mục mới có tên **Book.** Nhấp chuột phải vào **thư mục Sách** và thêm **Chế độ xem dao cạo** có tên là Index.cshtml.

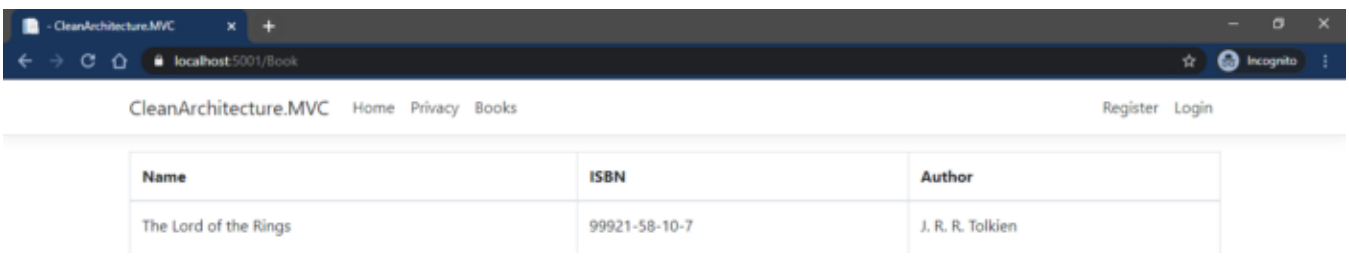


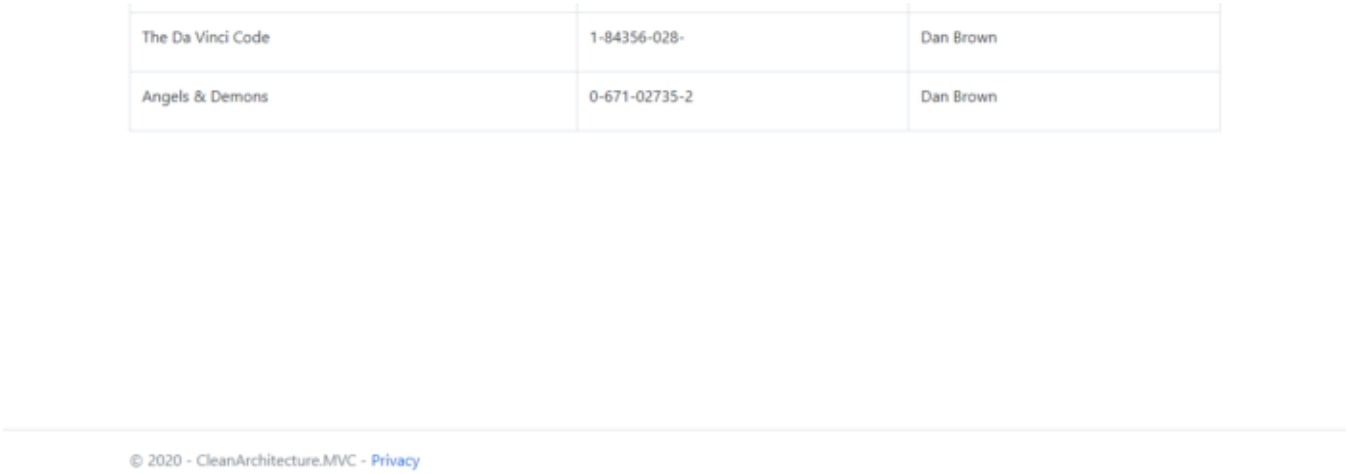


Trong **Index.cshtml, hãy** thêm đoạn mã sau, đoạn mã này sẽ lấy BookViewModel và lặp qua mọi mục (sách) trong mô hình và hiển thị nó trong một bảng.

Sau đó, truy cập **_Layout.cshtml** bên dưới **Chế độ xem / Đã chia sẻ** và thêm một liên kết điều hướng khác vào Chế độ xem mà chúng tôi vừa tạo. Lưu ý rằng nó trỏ đến bộ điều khiển **Sách** và hành động **Chỉ mục** .

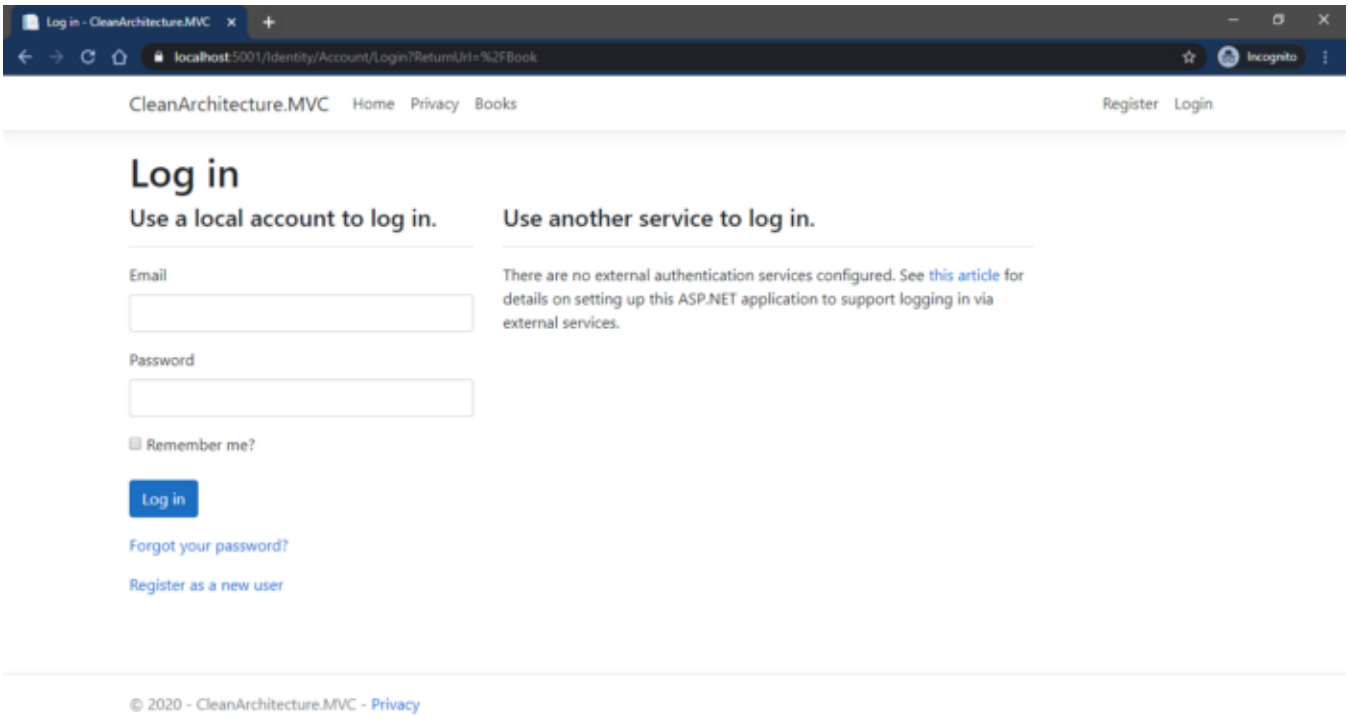
Bây giờ bạn có thể chạy dự án MVC và điều hướng đến https: // localhost: 5001 / Bookđể xem Sách!





Nhưng chúng tôi chưa đăng nhập, vì vậy việc cho phép người dùng xem dữ liệu mà không cần xác thực là một vấn đề, rất dễ giải quyết.

Thêm thuộc tính **[Authorize]** vào phương thức Index () vào chính bộ điều khiển và sau đó cố gắng điều hướng đến https: // localhost: 5001 / Đặt lại.



Bạn sẽ được yêu cầu đăng nhập. Điều bạn cũng có thể làm là đặt các điểm ngắt vào các dịch vụ, kho lưu trữ mà chúng tôi đã tạo và xem chúng hoạt động như thế nào trong thời gian chạy.

Vậy là xong, Cảm ơn bạn đã đến đây. Đây chỉ là một hướng dẫn giới thiệu về cách thiết lập mọi thứ, có rất nhiều cách khác nhau để triển khai kiến trúc sạch nhưng với các khái niệm cốt lõi mà tôi đã thảo luận ở đây. Vì vậy, lần sau khi bạn xem hướng dẫn mở rộng hơn, bạn sẽ không bị lạc.

Nếu bạn muốn đi xa hơn từ điều này, tôi khuyên bạn nên xem video này của JASON TAYLOR hoặc đọc bài viết tuyệt vời của anh ấy về Kiến trúc sạch .

Kiến trúc sạch sẽ Thực hiện Net Core 3 Dotnet Core