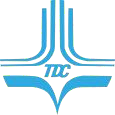
**TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC**

Khoa Công Nghệ Thông Tin

**CHUYÊN ĐỀ PHÁT TRIỂN WEB 1 | HKI – [2020 – 2021]**

**PRJ – REPORT**

**TÌM HIỂU ASP.NET Core**

**Nhóm M:**

Doãn Hồng Thăng – 18211TT4557

Nguyễn Y Trinh – 18211TT4551

Ngô Đình Trí – 18211TT2467

**BẢNG PHÂN CHIA CÔNG VIỆC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MSSV** | **Họ & Tên** | **Công việc** | **Hoành thành** | **Đánh giá** |
| 18211TT4557 | Doãn Hồng Thăng | -Hướng dẫn gửi email MailKit | 100% | Hoành thành |
| -Hướng dẫn tích hợp Api login with Google |
| -Giới thiệu về UserIdentity và User Manager |
| -Giới thiệu về Identity Authentication |
| -Hướng dẫn thiết lập Identity |
| 18211TT2467 | Ngô Đình Trí | -Giới thiệu về EntityFramework Core | 100% | Hoành thành |
| -Giới thiệu về Migration |
| -Giới thiệu về Submit form |
| -Giới thiệu về truy vấn dữ liệu bằng LINQ |
| -Hướng dẫn tích hợp SASS |
| 18211TT4551 | Nguyễn Y Trinh | -Cài đặt ASP.NET Core | 100% | Hoành thành |
| -Cài đặt SDK cho ASP.NET Core 3.1 |
| -Cách khởi tạo project |
| -Giới thiệu các giao thức web application |
| -Giới thiệu mô hình MVC |
| -Mô tả cấu trúc main có trong ASP.NET Core |
| -Giới thiệu Middleware trong ASP.NET Core |
| -Giới thiệu Routing trong ASP.NET Core |
| -Giới thiệu về Model Validation |

**MỤC LỤC**

[CHƯƠNG 1. Giới thiệu ASP.NET Core cơ bản 1](#_Toc55790654)

[CHƯƠNG 2. Cài đặt ASP.NET Core và một số thủ thuật nhỏ 1](#_Toc55790655)

[2.1. Cài đặt Microsoft Visual Studio 1](#_Toc55790656)

[2.2. Cài đặt SDK Core 3.1 2](#_Toc55790657)

[2.3. Cách khởi tạo project 3](#_Toc55790658)

[2.4. Giới thiệu mô hình MVC 6](#_Toc55790659)

[2.5. Mô tả cấu trúc main có trong ASP.NET Core 7](#_Toc55790660)

[2.6. Giới thiệu Middleware trong ASP.NET Core 7](#_Toc55790661)

[2.7. Giới thiệu Routing trong ASP.NET Core 9](#_Toc55790662)

[2.8. Giới thiệu về Model Validation 14](#_Toc55790663)

[2.9. Tích hợp SASS 15](#_Toc55790664)

[CHƯƠNG 3. Khái niệm căn bản trong ASP.NET Core 19](#_Toc55790665)

[3.1. Giới thiệu về EntityFramework Core 19](#_Toc55790666)

[3.2. Giới thiệu về Migration 24](#_Toc55790667)

[3.3. Giới thiệu về Submit form 28](#_Toc55790668)

[3.4. Giới thiệu về truy vấn dữ liệu bằng LINQ 30](#_Toc55790669)

[3.5. Giới thiệu về Identity Authentication 31](#_Toc55790670)

[3.6. Hướng dẫn thiết lập Identity 32](#_Toc55790671)

[3.7. Giới thiệu về UserIdentity và User Manager 32](#_Toc55790672)

[CHƯƠNG 4. Khái niệm nâng cao trong ASP.NET Core 33](#_Toc55790673)

[4.1. Hướng dẫn gửi email MailKit 33](#_Toc55790674)

[4.2. Hướng dẫn tích hợp API login with Google 34](#_Toc55790675)

**DANH MỤC HÌNH ẢNH**

[Hình 1. Trang download Microsoft Visual Studio 1](#_Toc55790773)

[Hình 2. ASP.NET and web development 2](#_Toc55790774)

[Hình 3. .NET Core cross-platform development 2](#_Toc55790775)

[Hình 4. Trang download .NET Core 2](#_Toc55790776)

[Hình 5. File dotnet-sdk-3.1 3](#_Toc55790777)

[Hình 6. Giao diện khởi tạo project của Microsoft Visual Studio. 3](#_Toc55790778)

[Hình 7. Tìm mục “asp.net web application” 3](#_Toc55790779)

[Hình 8. Khởi tạo mới project ASP.NET Core Web Application 4](#_Toc55790780)

[Hình 9. Các project templates của project ASP.NET Core Web Application 4](#_Toc55790781)

[Hình 10. Project vừa được tạo mới 5](#_Toc55790782)

[Hình 11. Mô hình MVC 6](#_Toc55790783)

[Hình 12. Code file Program.cs 7](#_Toc55790784)

[Hình 13. Sơ đồ Middleware trong ASP.NET Core 8](#_Toc55790785)

[Hình 14. Demo về middleware 8](#_Toc55790786)

[Hình 15. Middleware có sẵn. 8](#_Toc55790787)

[Hình 16. Mô hình Routing trong ASP.NET Core 9](#_Toc55790788)

[Hình 17. Routing middleware trong ASP.NET Core 9](#_Toc55790789)

[Hình 18. Trang chủ mới được tạo 10](#_Toc55790790)

[Hình 19. Controllers và Views 10](#_Toc55790791)

[Hình 20. Tạo “New Item” 11](#_Toc55790792)

[Hình 21. Chọn kiểu Controller Class – Empty 11](#_Toc55790793)

[Hình 22. Code file Controller, IActionResult trả về View(). 12](#_Toc55790794)

[Hình 23. Add View() được trả về 12](#_Toc55790795)

[Hình 24. Razor View 13](#_Toc55790796)

[Hình 25. Cấu hình Razor View 13](#_Toc55790797)

[Hình 26. . Index.cshtml được trả về 13](#_Toc55790798)

[Hình 27. Trang Web vừa được tạo 14](#_Toc55790799)

[Hình 28. Mô hình Model Validation 14](#_Toc55790800)

[Hình 29. wwwroot 15](#_Toc55790801)

[Hình 30. Chọn loại SASS để tạo mới. 15](#_Toc55790802)

[Hình 31. Web Compiler 16](#_Toc55790803)

[Hình 32. Compile .scss 🡪 .css 16](#_Toc55790804)

[Hình 33. Kết quả sau khi compile 16](#_Toc55790805)

[Hình 34. Thêm .css 17](#_Toc55790806)

[Hình 35. Index.cshtml 17](#_Toc55790807)

[Hình 36. File SASS 17](#_Toc55790808)

[Hình 37. Kết quả đạt được 18](#_Toc55790809)

[Hình 38. Database SQL Demo 19](#_Toc55790810)

[Hình 39. Các thao tác truy cập Manage NuGet Packages for Solution… 20](#_Toc55790811)

[Hình 40. Các NuGet Package hỗ trợ EF(Core) 20](#_Toc55790812)

[Hình 41. Cấu hình kết nối với Database 21](#_Toc55790813)

[Hình 42. Tạo một class ApplicationDbContext 21](#_Toc55790814)

[Hình 43. Kế thừa DbContext. Sau đó nhấn Alt + Enter để sử dụng thư viện 22](#_Toc55790815)

[Hình 44. Tiếp tục trỏ vào DbContext. Rồi Alt + Enter 🡪 Generate all. Tạo constructors. 22](#_Toc55790816)

[Hình 45. DbContextOptions<ApplicationDbContext> options 23](#_Toc55790817)

[Hình 46. Báo lỗi cần thêm thư viện. 23](#_Toc55790818)

[Hình 47. Trỏ vào [Key], Alt+Enter sau đó thêm thư viện vào. 23](#_Toc55790819)

[Hình 48. Đối tượng Product 24](#_Toc55790820)

[Hình 49. Vào lại ApplicationDbContext ánh xạ Product mới tạo vào CSDL 24](#_Toc55790821)

[Hình 50. Kết nối Database thông qua appsettings.json 24](#_Toc55790822)

[Hình 51. Microsoft.EnityFrameworkCore.Tools 25](#_Toc55790823)

[Hình 52. Tools 🡪 NuGet Package Manager 🡪 Package Manager Console 25](#_Toc55790824)

[Hình 53. Thêm Migration. 25](#_Toc55790825)

[Hình 54. Thông báo tạo Migration thành công 25](#_Toc55790826)

[Hình 55. Migration “CreateDBContext” được tạo 26](#_Toc55790827)

[Hình 56. Thêm thuộc tính Price vào đối tượng Product 26](#_Toc55790828)

[Hình 57. Cập nhật thuộc tính Price của đối tượng Product 27](#_Toc55790829)

[Hình 58. Migration UpdateProduct được tạo 27](#_Toc55790830)

[Hình 59. Cập nhật thành công lên SQL 27](#_Toc55790831)

[Hình 60. Dữ liệu phía SQL được tạo thành công 28](#_Toc55790832)

[Hình 61. Login Controllers. 28](#_Toc55790833)

[Hình 62. Tạo form trong Login.cshtml 29](#_Toc55790834)

[Hình 63. Đặt BreakPoint trong Login Controller 29](#_Toc55790835)

[Hình 64. Trang login sau khi chạy Project. 30](#_Toc55790836)

[Hình 65. Kết quả khi gửi đi. 30](#_Toc55790837)

[Hình 66. Đối tượng product viết bằng LINQ 31](#_Toc55790838)

[Hình 67. Thêm sản phẩm 31](#_Toc55790839)

[Hình 68. Demo 31](#_Toc55790840)

[Hình 69. Microsoft.AspNetCore.Identity.EntityFrameworkCore 32](#_Toc55790841)

[Hình 70. Custom mật khẩu cho phía identity. 32](#_Toc55790842)

[Hình 71. Khởi tạo đối tượng quản lý người dùng và đăng nhập 33](#_Toc55790843)

[Hình 72. Khởi tạo thông tin user trong database và đăng nhập 33](#_Toc55790844)

[Hình 73. Mô tả hình thức gửi message qua email 34](#_Toc55790845)

[Hình 74. Đăng nhập tài khoản Google 35](#_Toc55790846)

[Hình 75. Lựa chọn project trong APIs Services Of Google 35](#_Toc55790847)

[Hình 76. Tạo mới Project trong APIs Services Of Google 36](#_Toc55790848)

[Hình 77. Đặt tên cho Project API 36](#_Toc55790849)

[Hình 78. Mở các thư viện có trong APIs Services Google 37](#_Toc55790850)

[Hình 79. Tìm Google plus API 37](#_Toc55790851)

[Hình 80. Tích hợp Google+ API 37](#_Toc55790852)

[Hình 81. Kích hoạt Google+API 38](#_Toc55790853)

[Hình 82. Hình hiển thị thành công sao khi dùng Google+ API 38](#_Toc55790854)

[Hình 83. Khởi tạo Oauth Client Id và Secret Id 39](#_Toc55790855)

[Hình 84. Xác nhận việc cho phép nhiều người dùng đăng nhập 39](#_Toc55790856)

[Hình 85. Copy tên project 40](#_Toc55790857)

[Hình 86. Điền Tên Project vào đây 40](#_Toc55790858)

[Hình 87. Lưu thông tin project API 41](#_Toc55790859)

[Hình 88. Scopes trong Services Of Google 41](#_Toc55790860)

[Hình 89. Thông tin về giới thiệu dịch vụ Google 42](#_Toc55790861)

[Hình 90. Quay trở lại trang thông tin API 42](#_Toc55790862)

[Hình 91. Khởi tạo Credentials cho API 43](#_Toc55790863)

[Hình 92. Khởi tạo Client Id 43](#_Toc55790864)

[Hình 93. Chọn loại project đang phát triển 43](#_Toc55790865)

[Hình 94. Chọn Web application 44](#_Toc55790866)

[Hình 95. Bước lấy App URL trong project 44](#_Toc55790867)

[Hình 96. Chọn Properties 45](#_Toc55790868)

[Hình 97. Copy SSL qua App URL 45](#_Toc55790869)

[Hình 98. Thiết lập URIs trong Oauth of Google Service 46](#_Toc55790870)

[Hình 99. Copy Client Id và Client Secret 46](#_Toc55790871)

[Hình 100. Thông tin về Account Oath 47](#_Toc55790872)

[Hình 101. Thông tin chi tiết về Client Id,Client Secret 47](#_Toc55790873)

[Hình 102. Mô tả Client Id và Client Secret 47](#_Toc55790874)

[Hình 103. Trang login 48](#_Toc55790875)

# 

# Giới thiệu ASP.NET Core cơ bản

ASP.NET Core là một tập hợp các thư viện quy chuẩn giống như một framework kiểu module để xây dựng ứng dụng web hoàn chỉnh. ASP.NET Core là mã nguồn mở, có hiệu suất cao và đa nền tảng.

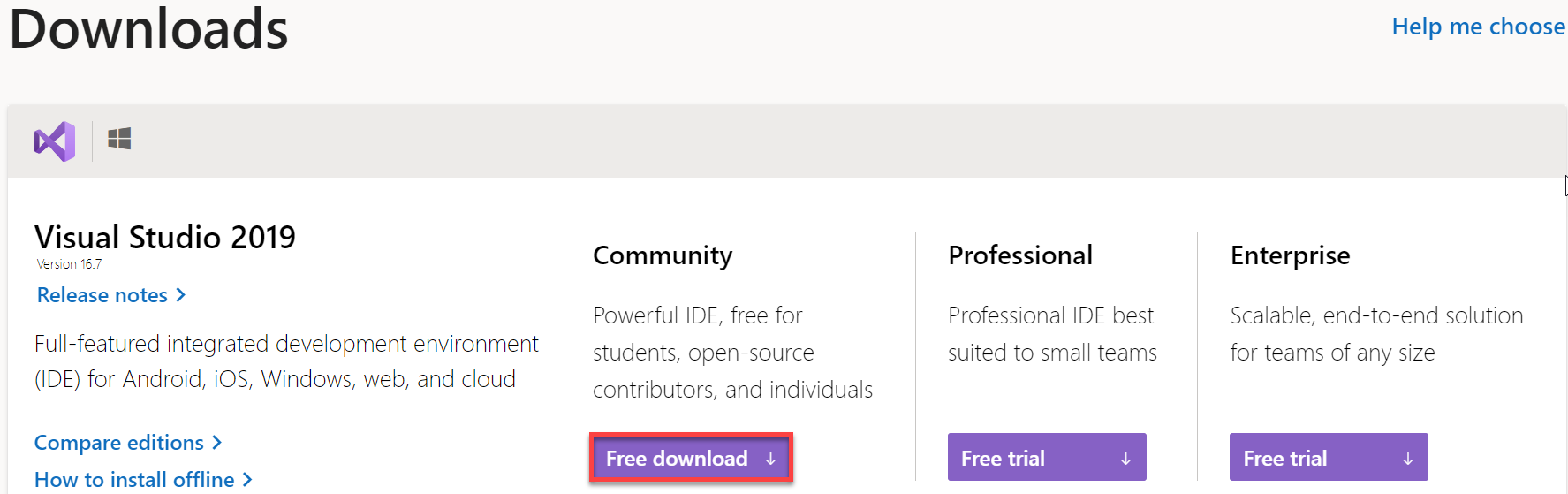
* **Ở mục II** : là các thủ thuật cài đặt và làm quen với ASP.NET Core 3.1 cơ bản
* **Ở mục III**: Giới thiệu về cách thiết lập và kết nối SQL cơ bản bằng EF Core, và các thao toán dẫn và truyền dữ liệu lên server
* **Ở mục IV**: Giới thiệu về MailKit và hướng dẫn tích hợp API external login of Google

# Cài đặt ASP.NET Core và một số thủ thuật nhỏ

## Cài đặt Microsoft Visual Studio

Để cài đặt Microsoft Visual Studio

Ta vào trang : <https://visualstudio.microsoft.com/downloads/>



1. Trang download Microsoft Visual Studio

Có 3 sự lựa chọn

**Community**: download free đối với cá nhân không phải là các nhà lập xây dựng sản phẩm thực tế và có sự hạn chế khi phát triển dự án nhỏ và lớn

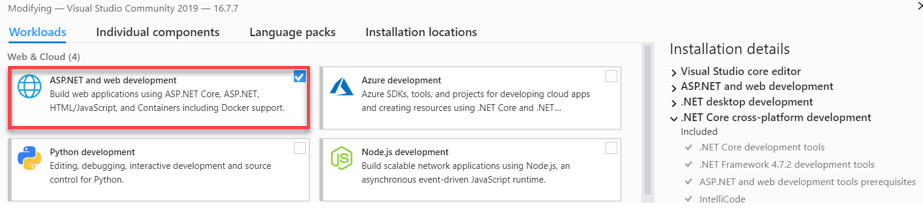
**Professional** :download trả phí đối với nhà sáng lập sản phẩm ở quy mô nhỏ. Còn sự hạn chế tuy không lớn

**Enterprise**: download trả phí đối với nhà sáng lập sản phẩm ở quy mô lớn.

-Sau khi tải về bản **Community** , chúng ta tiến hành cài đặt bình thường.

Sau đó khi hộp thoại Visual Studio Installer hiện lên ở phần Workloads trong Visual Studio Installer. Chúng ta cần cài 2 thứ:

* ASP.NET and web development



1. ASP.NET and web development

* .NET Core cross-platform development

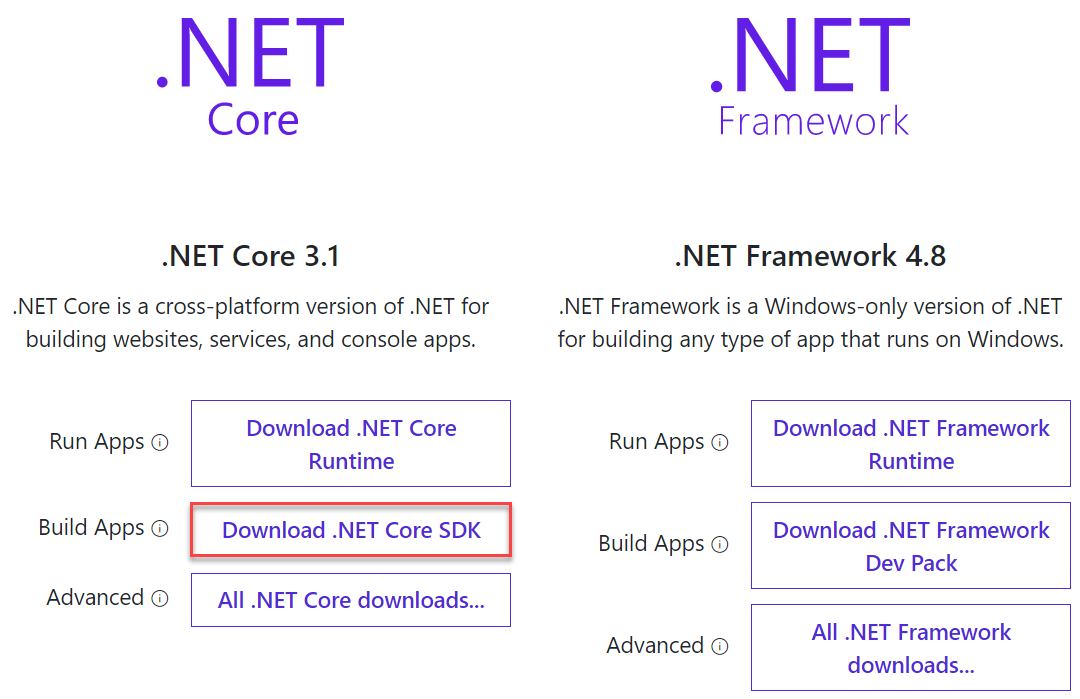


1. .NET Core cross-platform development

🡪Khi cài đặt xong đi tiếp đến Cài đặt SDK ở bên dưới.

## Cài đặt SDK Core 3.1

Vào trang : <https://dotnet.microsoft.com/download>



1. Trang download .NET Core

Sau khi tải xuống ta được hình bên dưới



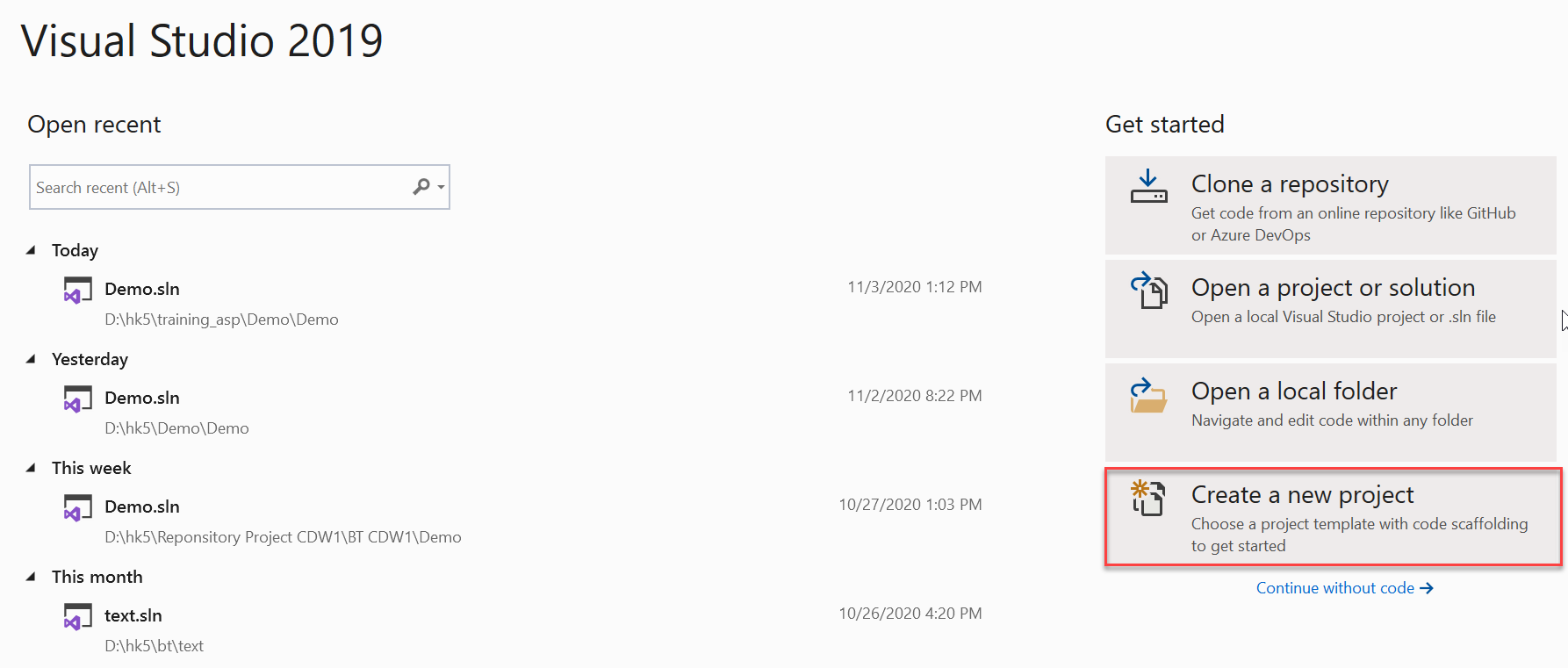
1. File dotnet-sdk-3.1

Tiếp theo nhấn và khởi chạy với đường dẫn lúc đầu khi cài đặt visual

## Cách khởi tạo project

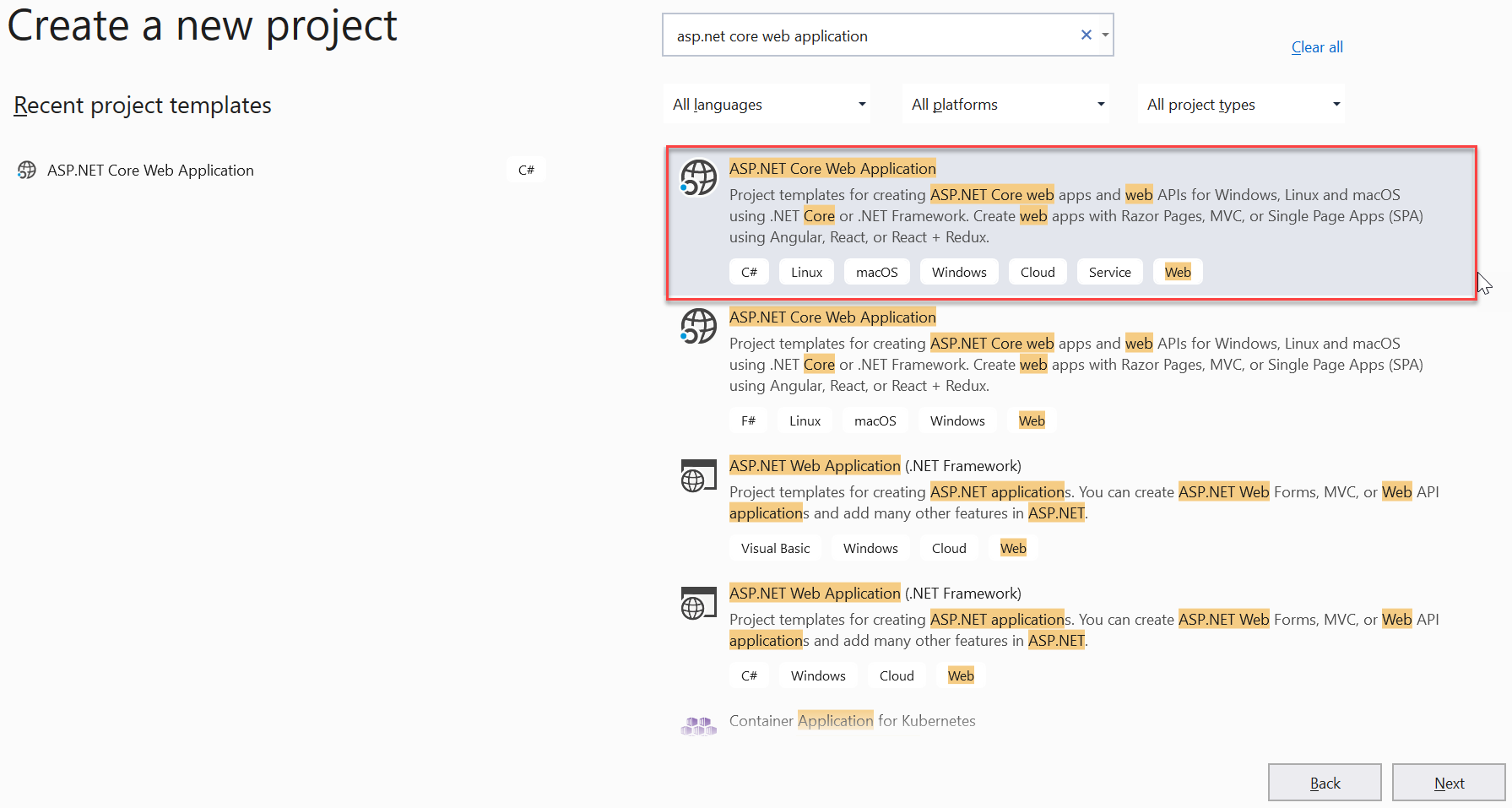
Mở giao diện Microsoft Visual Studio lên

**B1:** Nhấn vào nút



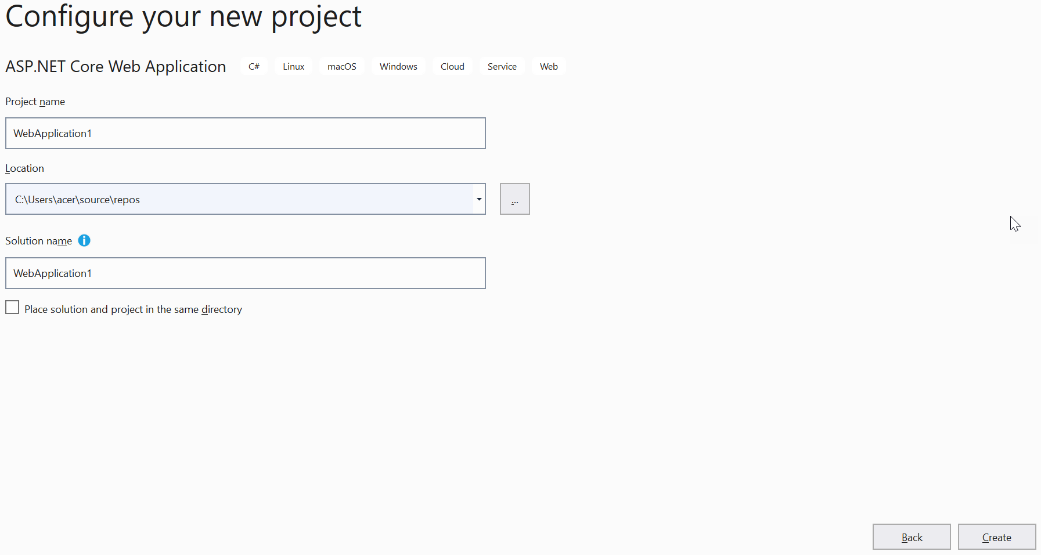
1. Giao diện khởi tạo project của Microsoft Visual Studio.

**B2:** Tìm kiếm “asp.net web application”



1. Tìm mục “asp.net web application”

**B3:** Khởi tạo name và đường dẫn



1. Khởi tạo mới project ASP.NET Core Web Application

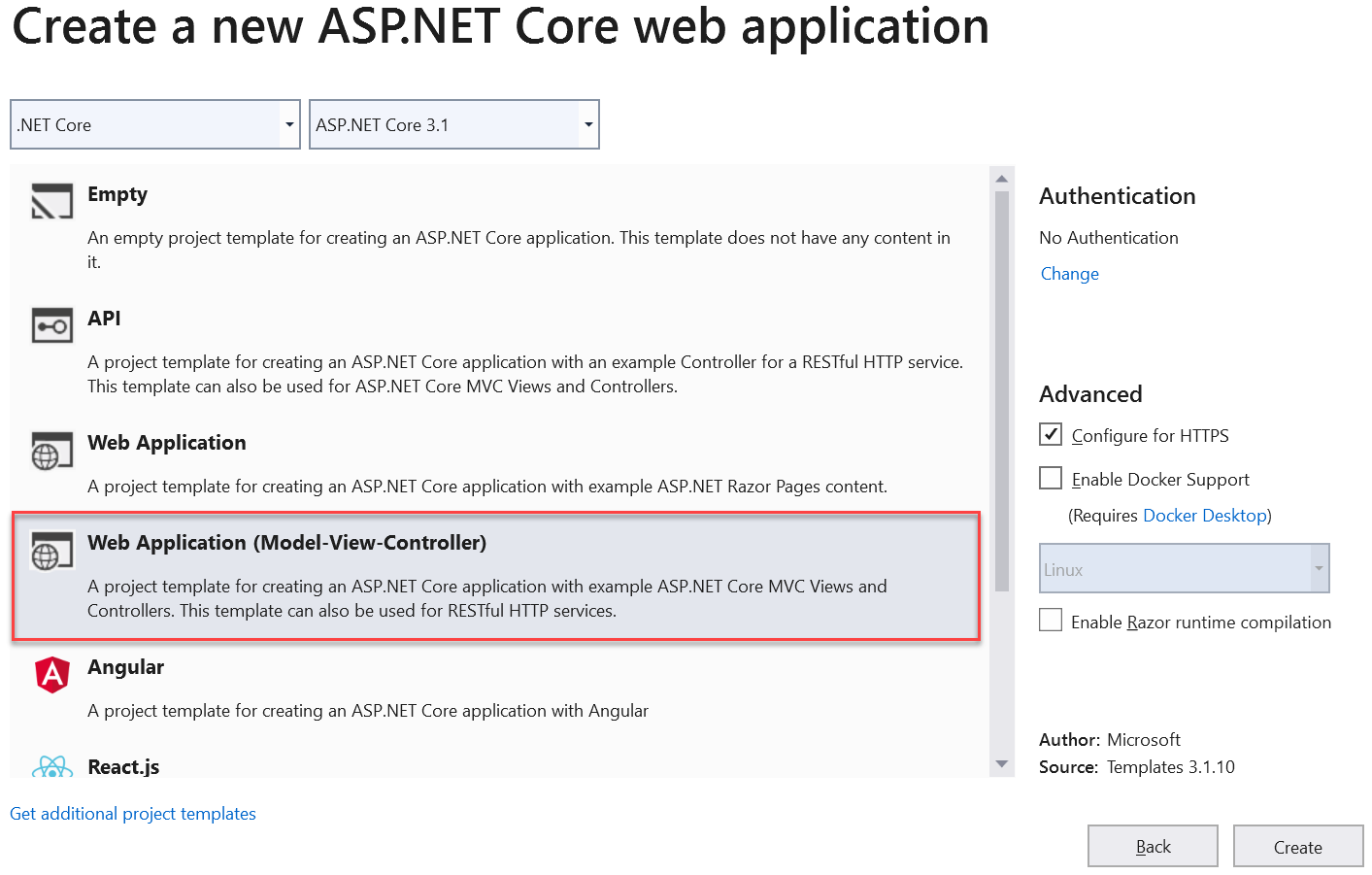
Project name : Khởi tạo lại tên project nếu muốn

Location: Đường dẫn lưu trữ của project

=> Sau khi thao tác xong 2 phần trên nhấn create

**B4:** Chọn Project Template

Bước tiếp theo về việc khởi tạo project.



1. Các project templates của project ASP.NET Core Web Application

Empty

-Khi khởi tạo mặc định nó sẽ không có controller, action ,model,Routing. Người dùng phải xây dựng lại theo cách của mình nếu muốn

=> Dành cho những ví dụ hoặc người thích sáng tạo theo cách của bản thân

API

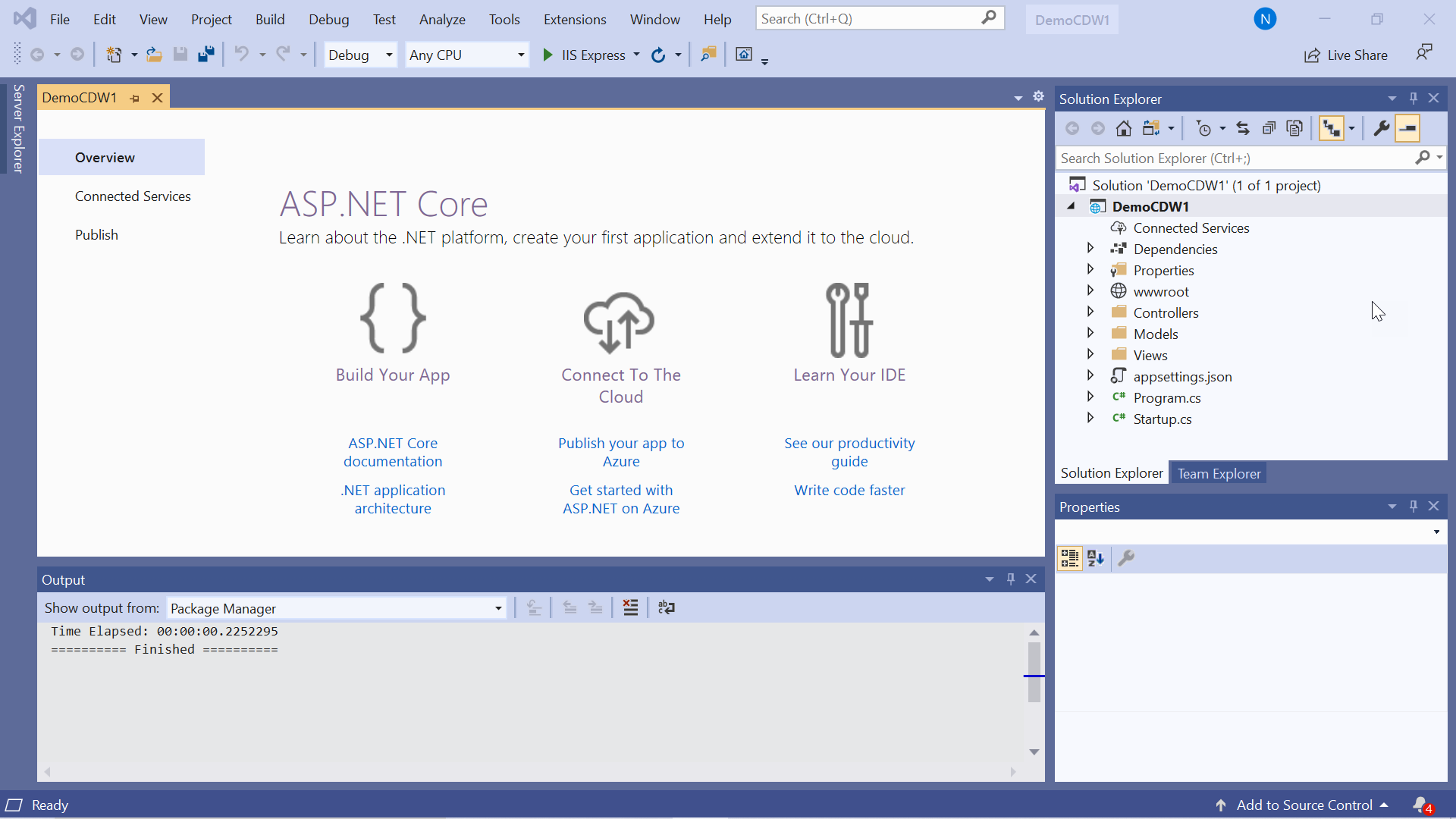
-Khi khởi tạo, mặc định project sẽ nhận được các thư viện liên quan đến API để khởi tạo API cho các framework như React,Angular,Vue...

Web Application(Model-View-Controller)

-Tạo ra model, View, Controller tức là một bản có cấu trúc mvc và routing hoàn chỉnh cho việc phát triển web

Nhấn chọn Web Application(Model-View-Controller) để tiếp tục.

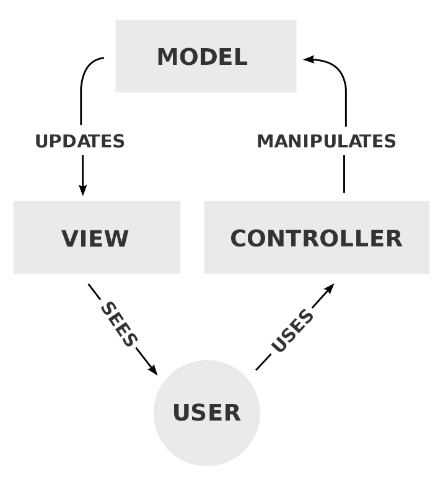
Sau khi project được tạo. Ta được project như sau:



1. Project vừa được tạo mới

## Giới thiệu mô hình MVC

Mô hình mvc được viết tắt bởi 3 từ model - view – controller. Mô hình được sử dụng trong kỹ thuật phần mềm.



1. Mô hình MVC

Model

Là phần chức năng lưu trữ toàn bộ dữ liệu, được kết nối với sql, để tạo các câu truy vấn, truy xuất và xử lý dữ liệu ... Là một cầu nối giữa view và controller

View

Đây là phần giao diện dành cho người sử dụng, nó chứa các template của trình duyệt của phần mềm web. Nơi người dùng lấy và xem các thông tin từ cơ sở dữ liệu

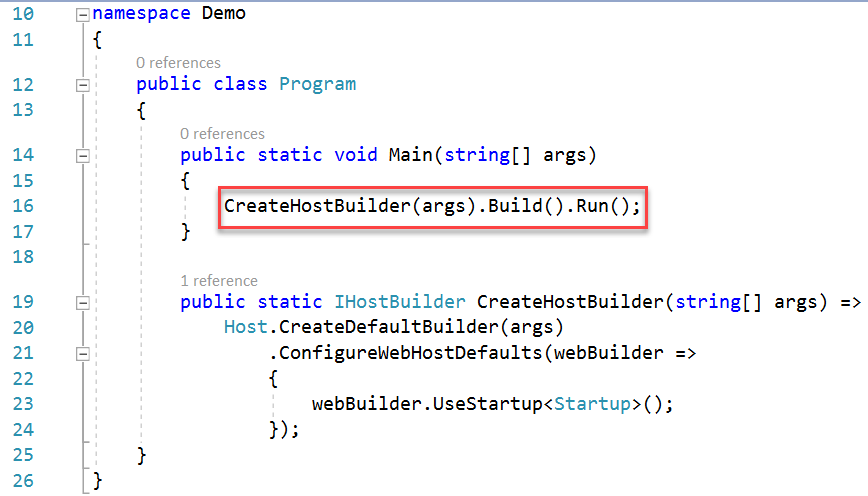
Controller

Bộ phần có nhiệm vụ xử lý yêu cầu người dùng thông qua view. Từ đó lấy ra các dữ liệu phù hợp nhất cho người dùng. Bên cạnh đó Controller có chức năng kết nối với model

Về ưu nhược điểm xem thêm tại: <https://monamedia.co/mvc-la-gi-ung-dung-cua-mo-hinh-mvc-trong-lap-trinh/>

## Mô tả cấu trúc main có trong ASP.NET Core

Program.cs



1. Code file Program.cs

Hàm main ở trên chứa một public static void main() method. Mặc định khi khởi tạo console application (ứng dụng giao diện điều khiển ) trong .net mặc định .Net framework biết rằng đây là nơi thực thi toàn bộ project

* Tại sao chúng ta lại phải có **main() method** ?

Ban đầu khi khởi tạo ASP.NET Core, nó sẽ đi vào hàm main với dòng

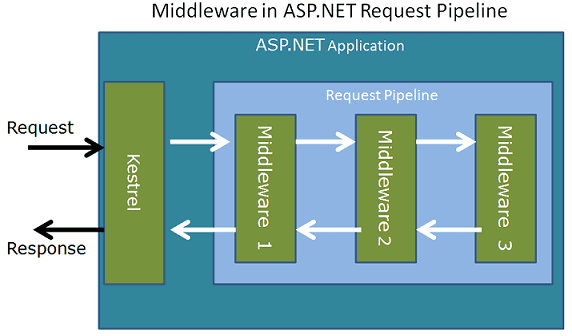
**CreateHostBuilder(args).Build().Run();**

**CreateHostBuilder() method** trả về một lệnh triển khai

một đối tượng thực thi giao diện IHostBuilder. Host là một lớp tĩnh có thể được sử dụng để tạo một phiên bản của IhostBuilder nhằm khởi tạo máy chủ nội bộ

## Giới thiệu Middleware trong ASP.NET Core

***-***Middleware là thành phần các lớp được thực thi theo yêu cầu của ứng dụng ASP.NET Core. Nó là đường ống xử lý yêu cầu “request pipeline”



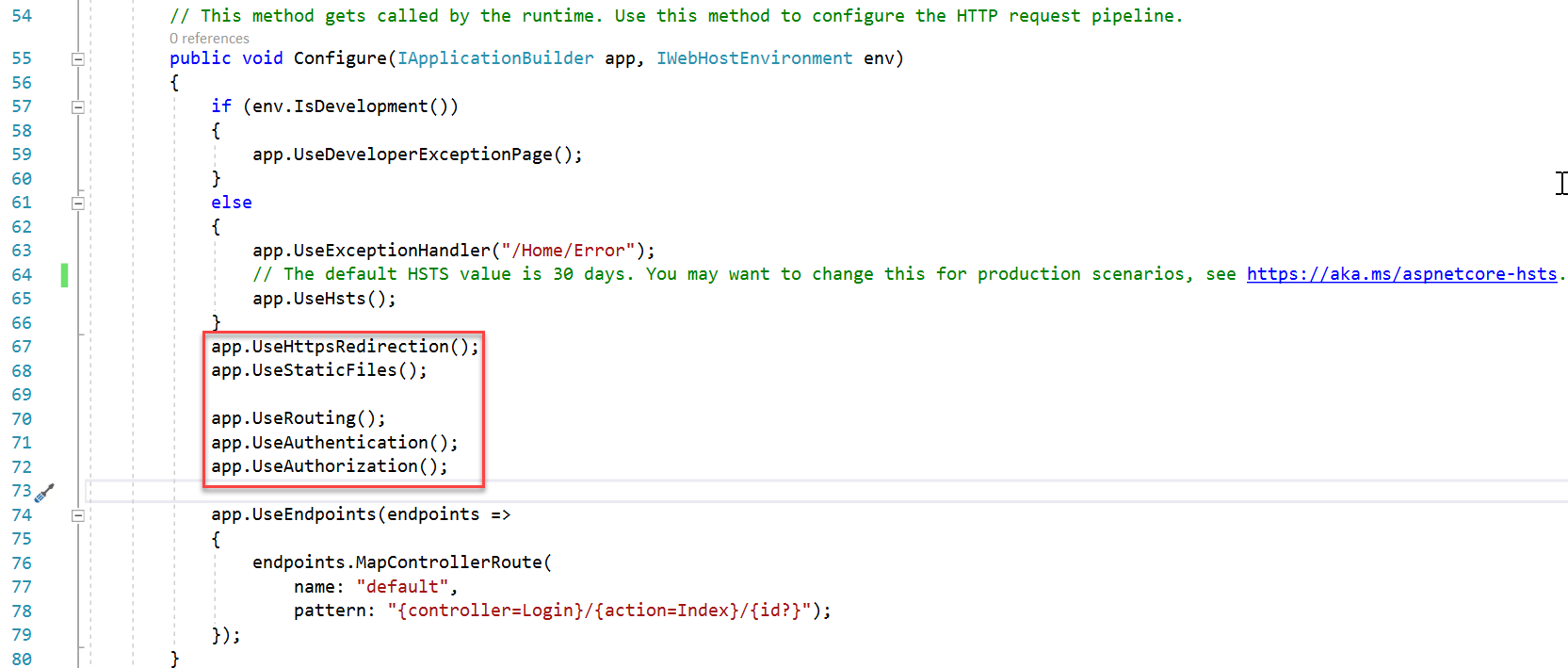
1. Sơ đồ Middleware trong ASP.NET Core

Có thể thấy ở đây, khi một request được đưa lên nó sẽ đi vào các middleware, khi đó các middlware khác nhau sẽ xử lý phần request và trả về cho thằng đứng sau, khi đến cuối middleware nó sẽ trả lại response



1. Demo về middleware

Khi một middleware được đưa lên, nó sẽ đi qua các app.run để trả về một response



1. Middleware có sẵn.

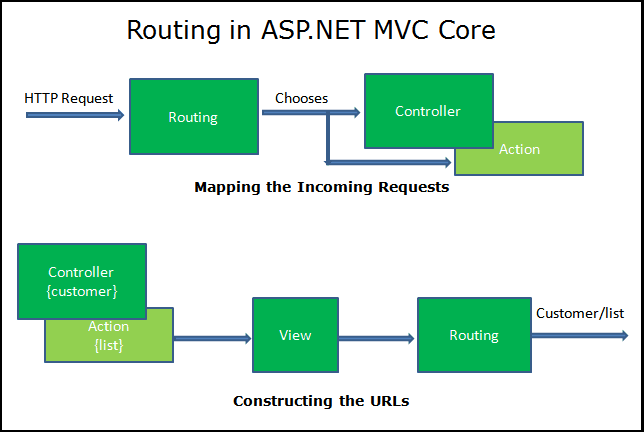
Thông thường với web application (mvc) nó có sẵn các middleware. Về các

phương thức Use() “UseHttpsRedirection(),UseStaticFiles(),UseRouting()...”

=> Nó là phương thức mở rộng, tương tự như run() ngoại trừ việc bao gồm các next để gọi middleware kế tiếp trong chuỗi

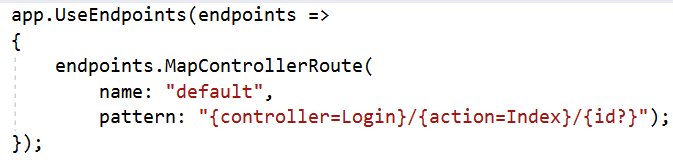
## Giới thiệu Routing trong ASP.NET Core

-Routing là quá trình xem xét các URL request gửi đến và chỉ đường cho đến các controller action. Tạo các URL đầu ra quá trình này đảm nhiệm bởi routing middleware trong ASP.NET Core.

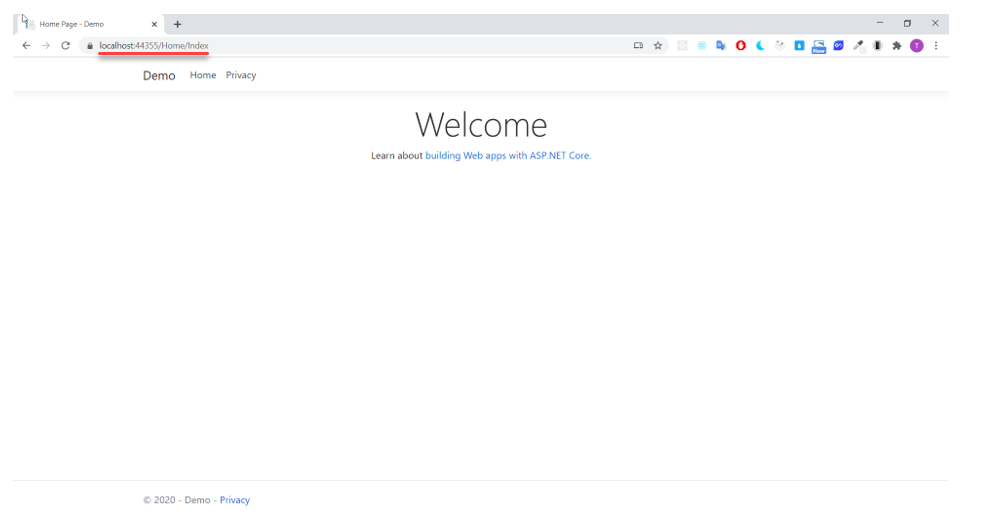


1. Mô hình Routing trong ASP.NET Core

Đối với Routing middleware trong ASP.NET Core như sau:



1. Routing middleware trong ASP.NET Core



1. Trang chủ mới được tạo

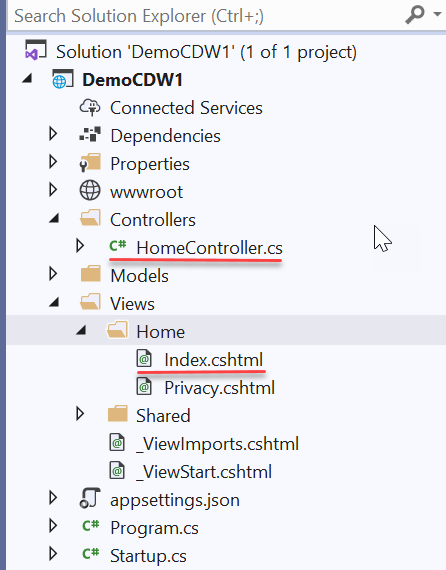
Khi nhấn /Home/Index => Nó tương tự như **pattern** ở phía trên

Controller = “Home”

Action = “Index”

Id = “” lúc này id chưa có nên nó sẽ là rỗng”

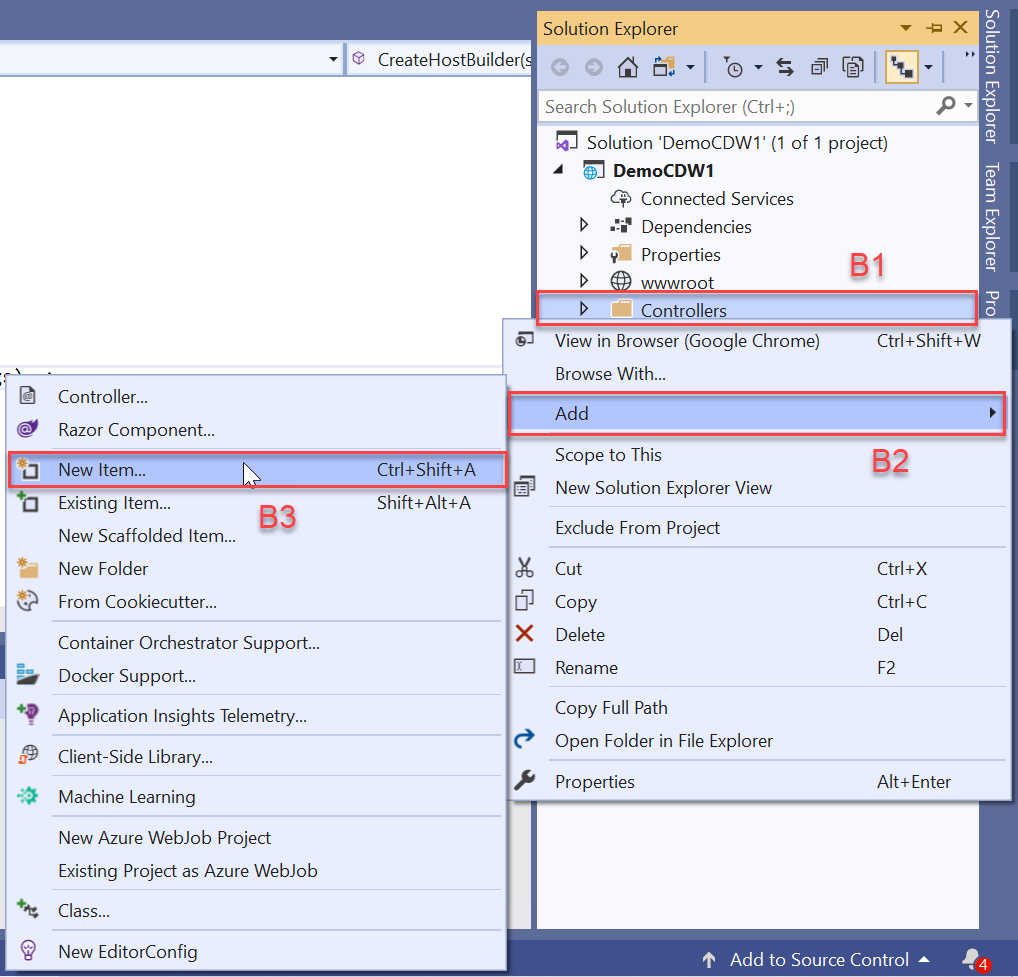
=> Sau khi gửi URL request lên ta thấy được routing trong middleware sẽ tìm các controller action



1. Controllers và Views

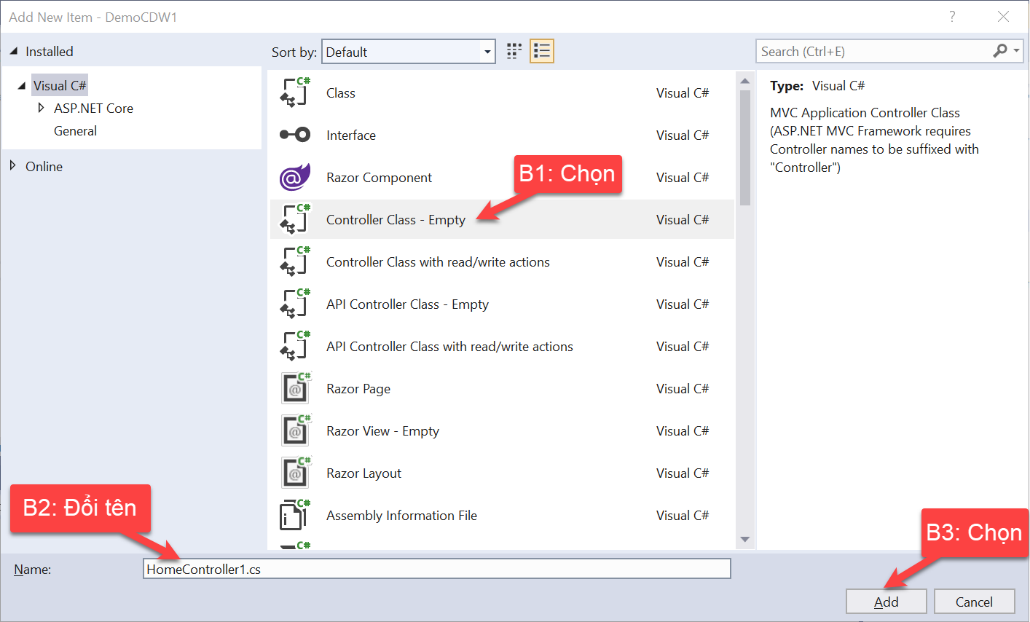
**Cách thêm 1 Routing.**

* Click chuột phải vào thư mục Controllers 🡪 Add 🡪 New Item. (Hoặc Ctrl + Shift + A)



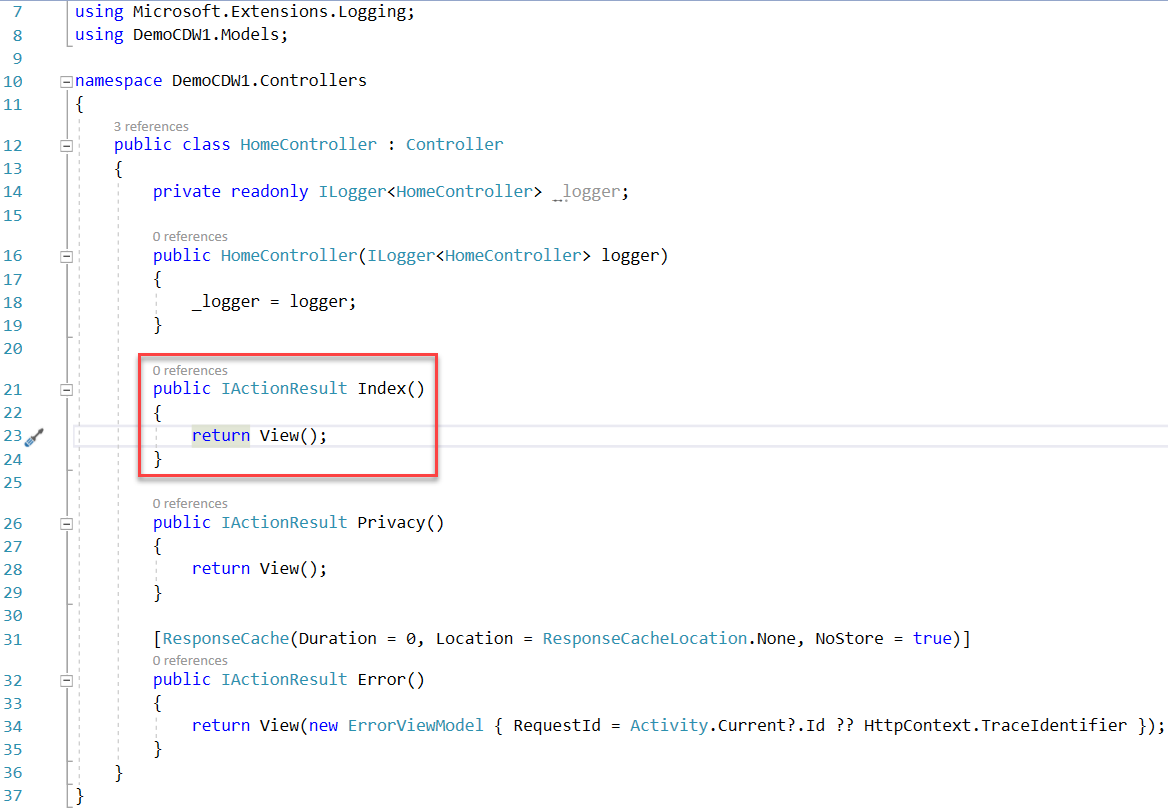
1. Tạo “New Item”

* Chọn Controller Class – Empty. Đặt tên file rồi nhấn “Add”.



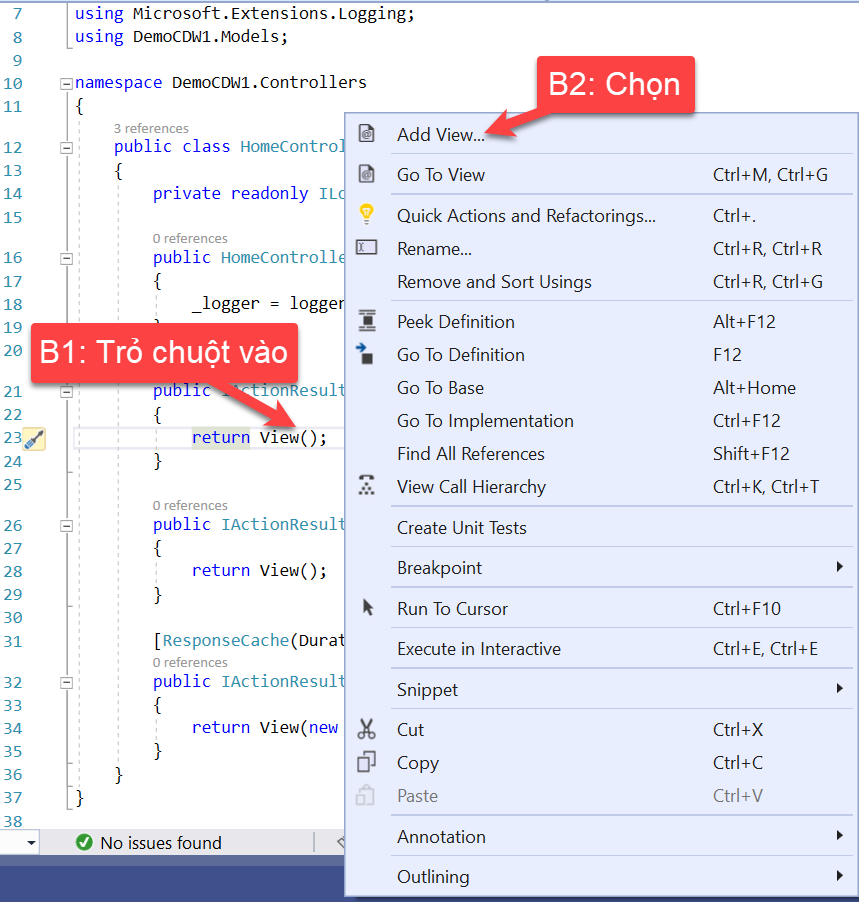
1. Chọn kiểu Controller Class – Empty

* Ta được 1 file Demo.cs như sau. Có phương thức IActionResult trả về View().



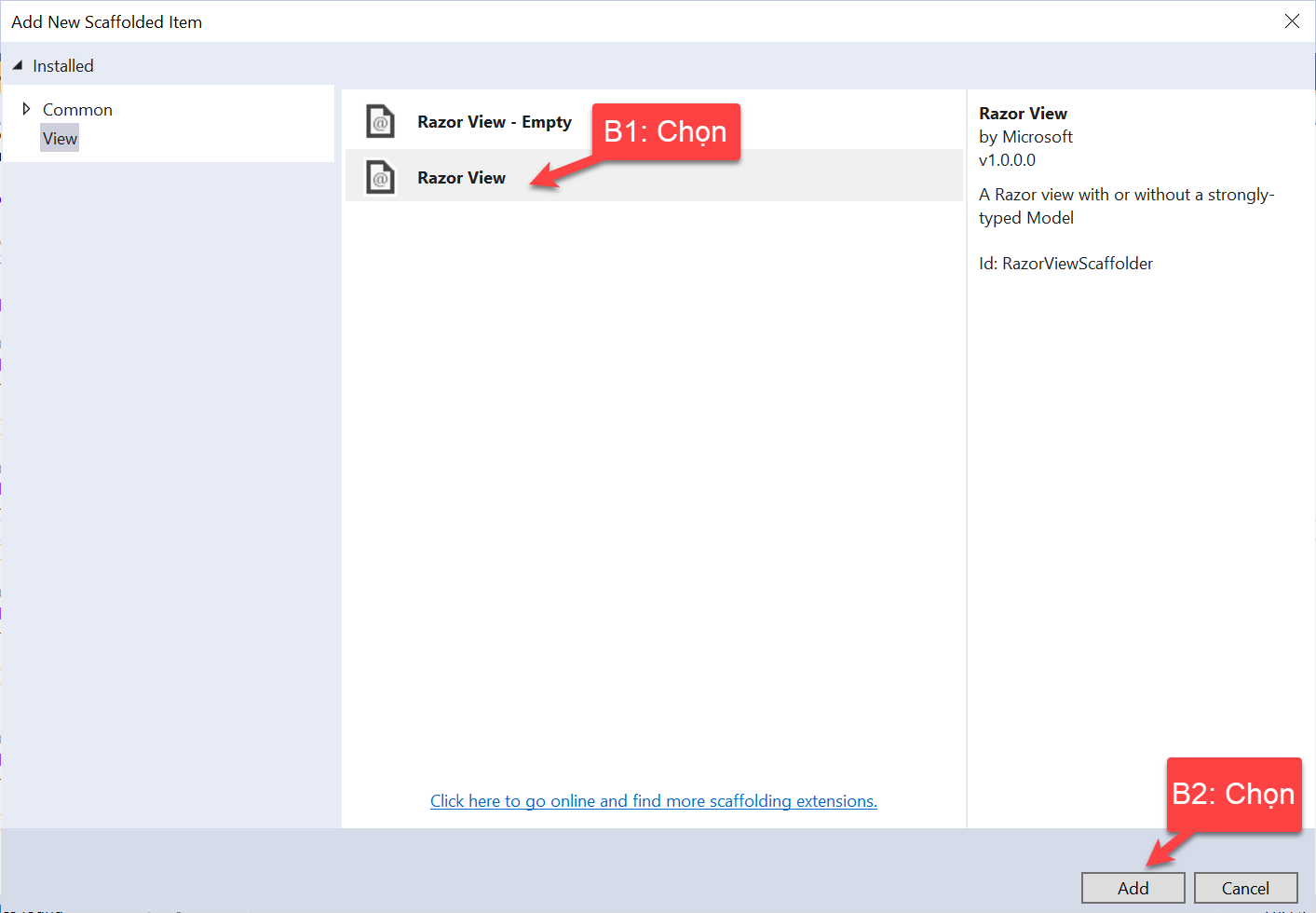
1. Code file Controller, IActionResult trả về View().

* Ta trỏ chuột vào View() và click chuột phải. Chọn Add View…



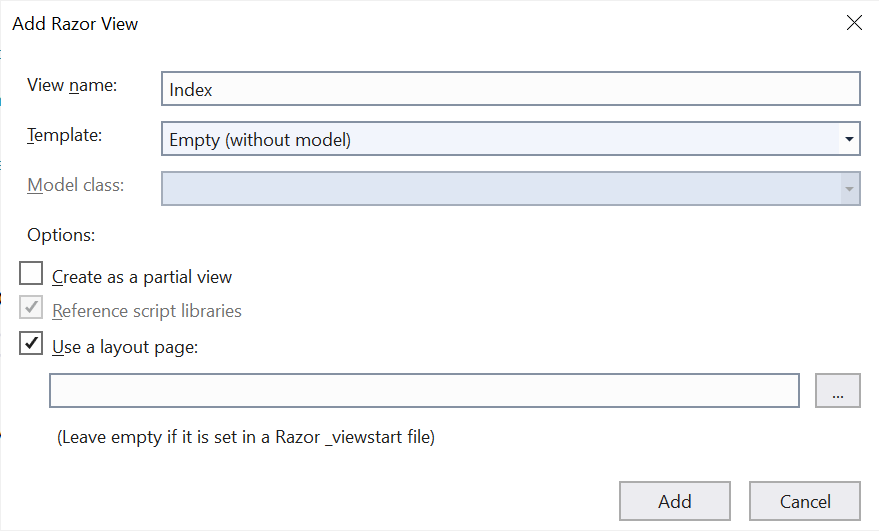
1. Add View() được trả về

* Chọn Razor View.



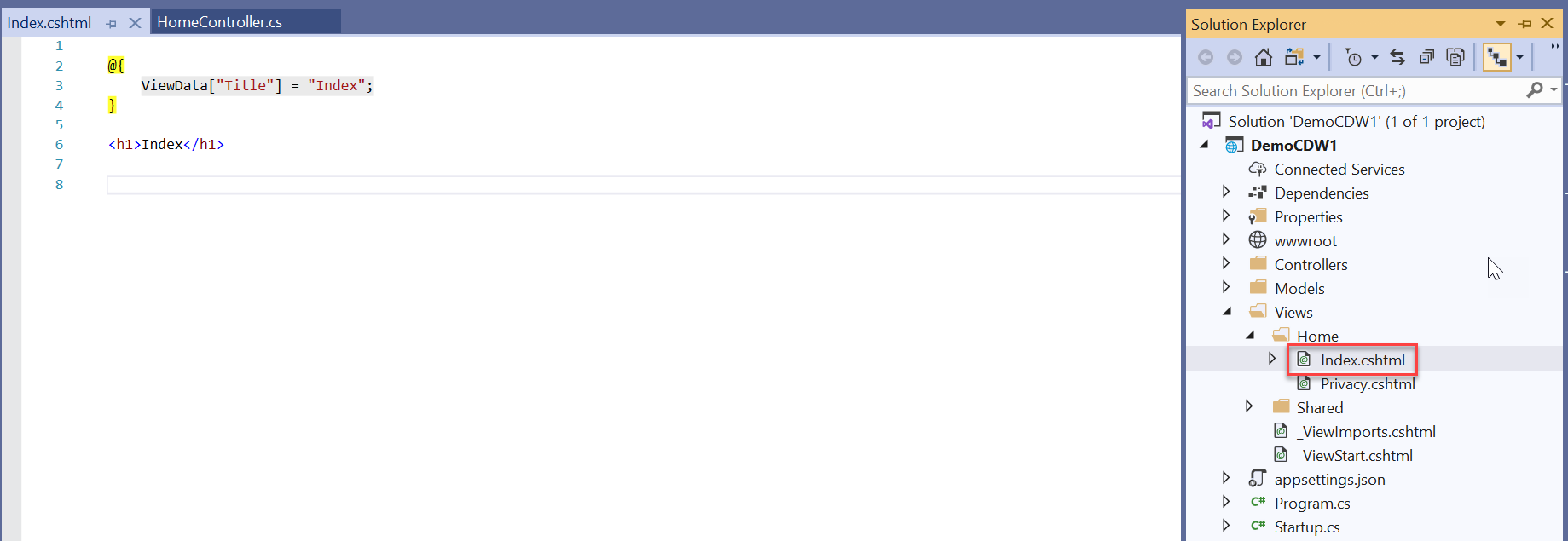
1. Razor View

* Tùy chỉnh một số thuộc tính. Ở đây mình để mặc định. Rồi nhấn “Add”



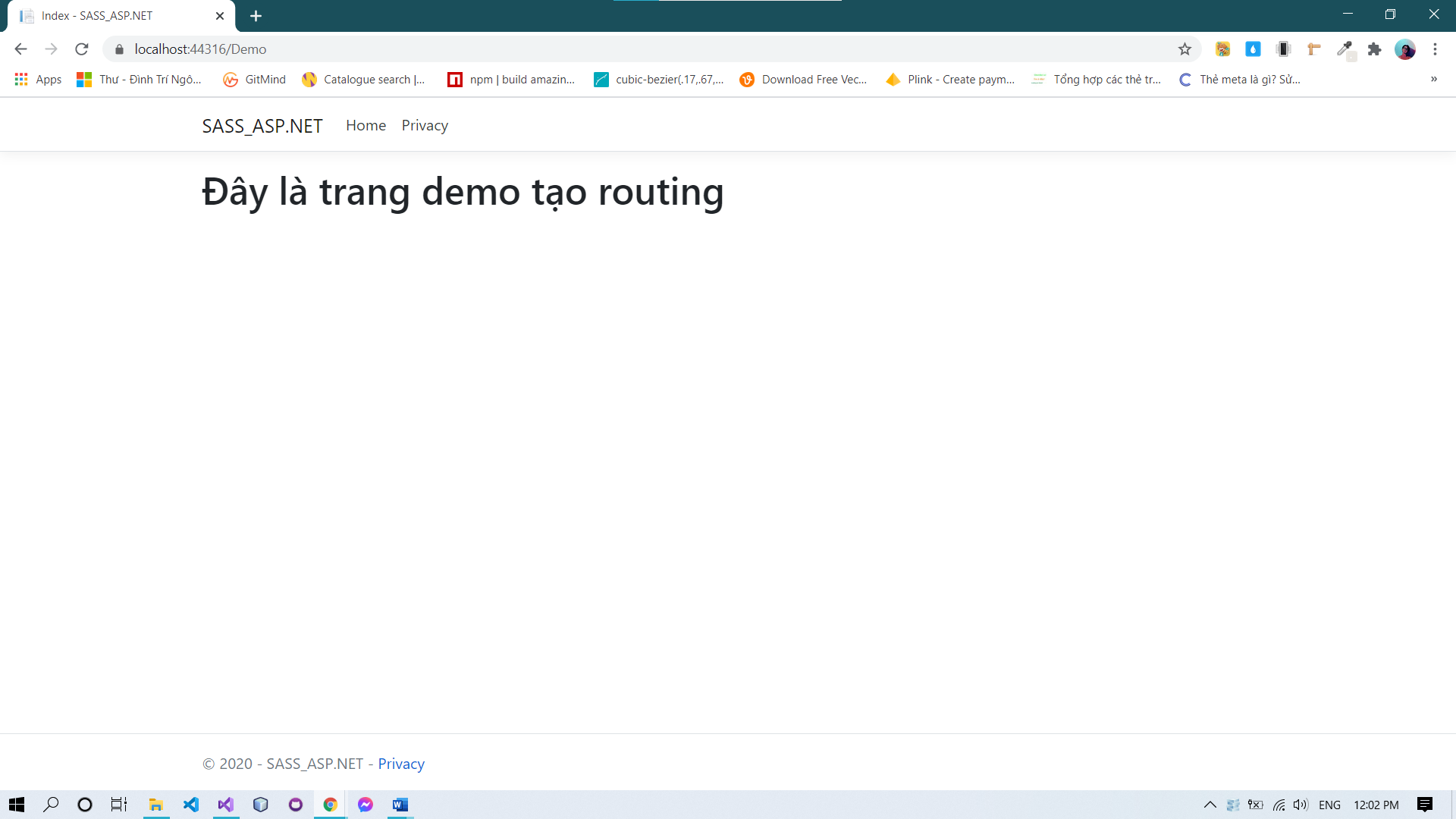
1. Cấu hình Razor View

* Một file Index.cshtml sẽ được tự động tạo trong folder Demo trong Views.



1. . Index.cshtml được trả về

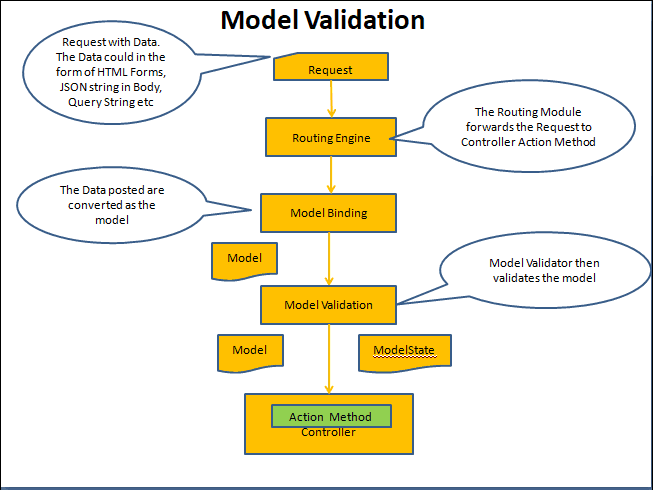
* Chạy chương trình vào đường dẫn localhost:xxxx/Demo. Ta sẽ truy cập tới trang vừa tạo.



1. Trang Web vừa được tạo

## Giới thiệu về Model Validation

Tự động được map data vào các tham số của action bởi Model Binder hợp lệ không. Quá trình kiểm tra này có thể được hoàn thành bởi client trước khi gửi lên server hoặc server kiểm tra sau khi nhận được từ client



1. Mô hình Model Validation

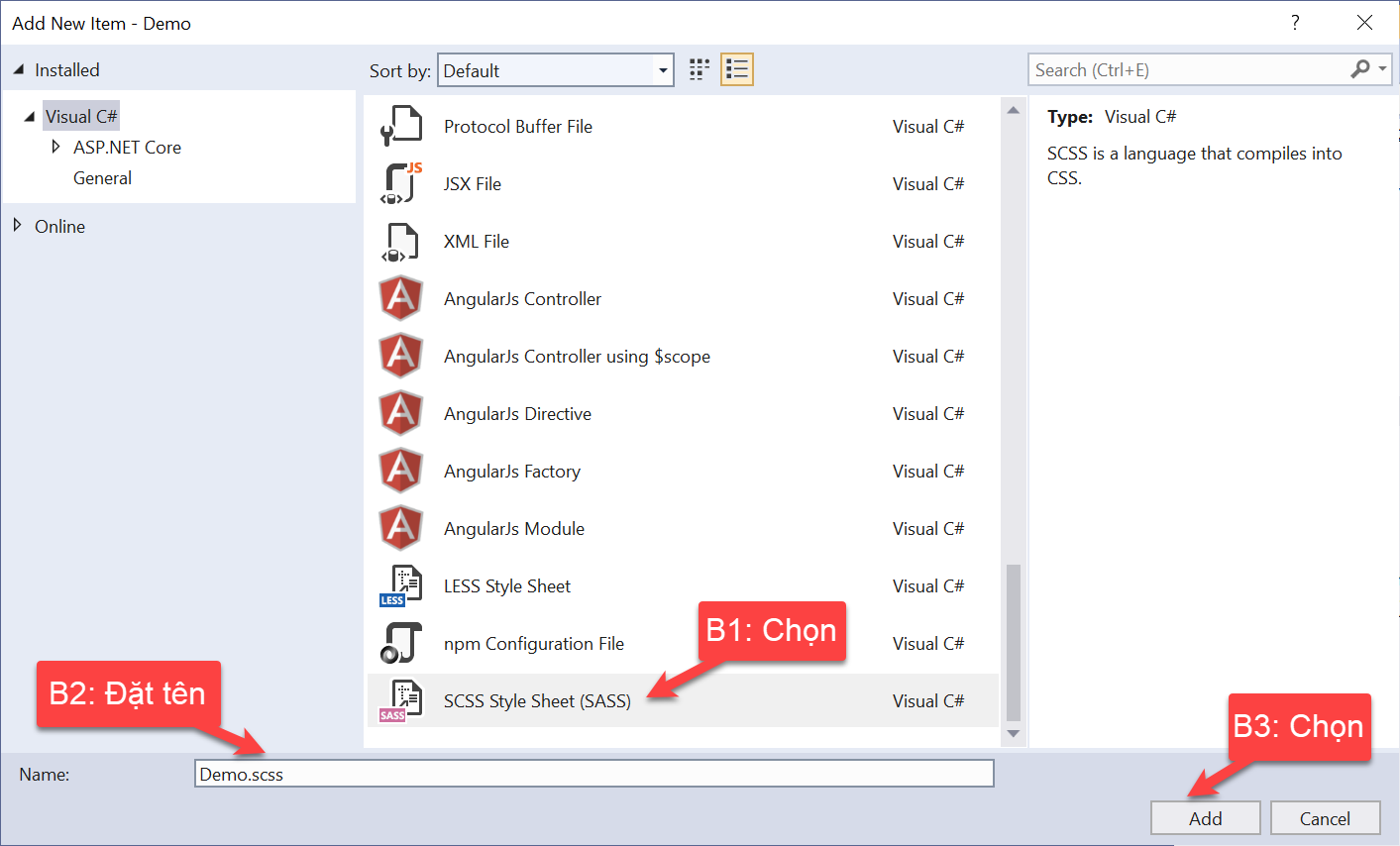
## Tích hợp SASS

* Tạo folder SASS trong wwwroot.



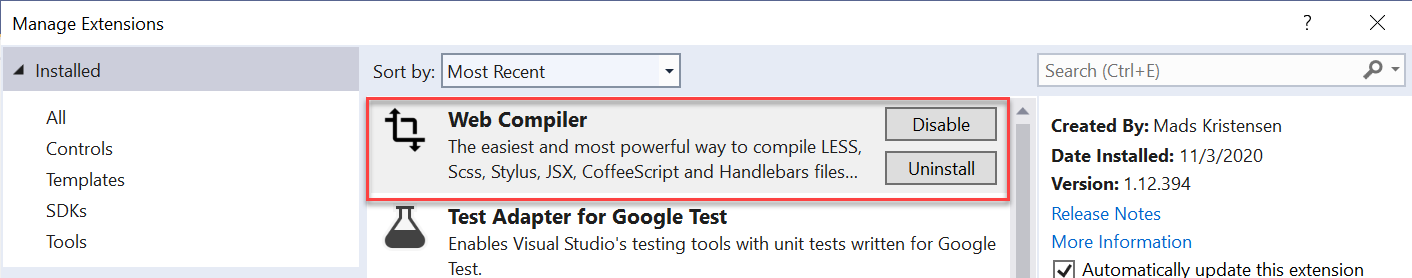
1. wwwroot

* Chọn thư mục sass rồi nhấn tổ hợp phím Ctrl + Shift + A để thêm item mới. Chọn SCSS Style Sheet (SASS). Đặt tên file rồi tạo mới.



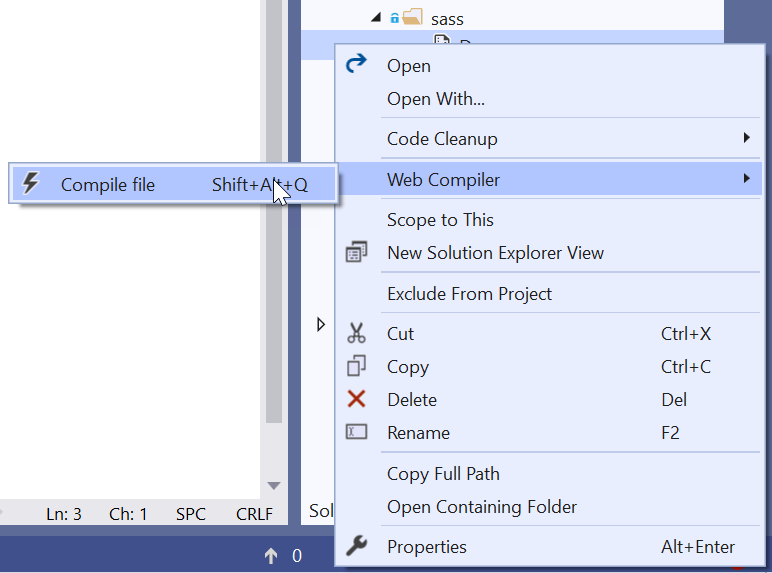
1. Chọn loại SASS để tạo mới.

* Cài extension “Web Compiler”. Có thể cài trong Manage Extensions của Microsoft Visual Studio. Hoặc tải online ở [link](https://marketplace.visualstudio.com/items?itemName=MadsKristensen.WebCompiler) này.



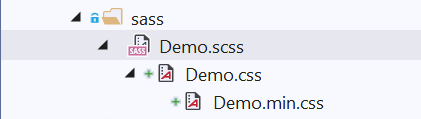
1. Web Compiler

* Chọn file .scss cần dùng rồi compile sang .css.



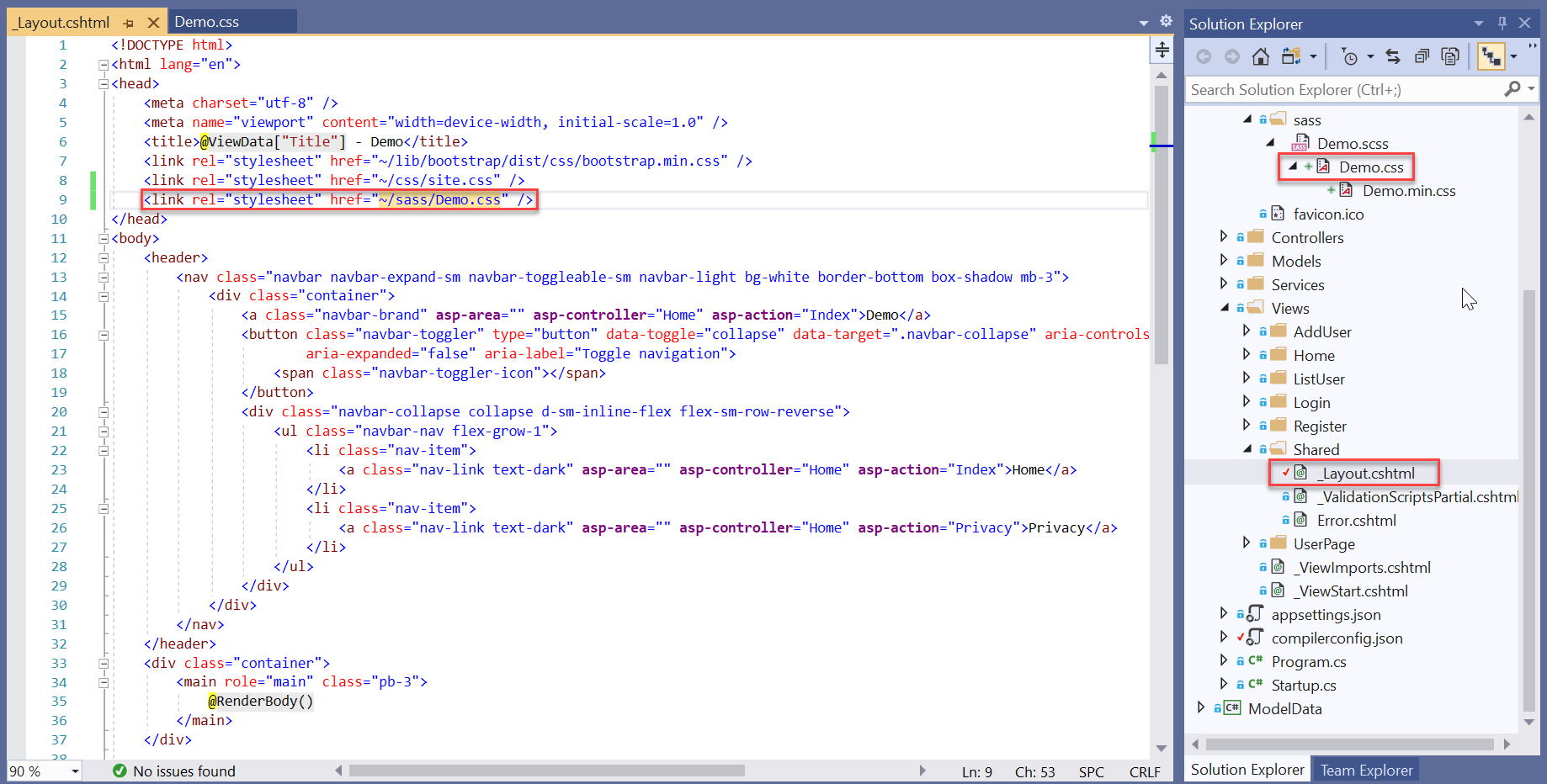
1. Compile .scss 🡪 .css

* Ta thu được file .css và .min.css. Extension “Web compiler” còn giúp đồng bộ hóa code scss sang css theo thời gian thực.



1. Kết quả sau khi compile

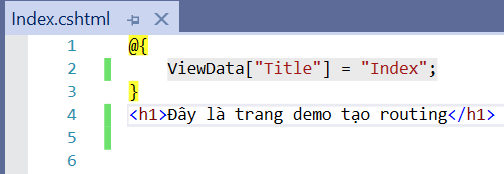
* Do ở đây trang demo của mình dùng file \_Layout.cshtml trong folder Shared. Nên ta sẽ bỏ file css vừa được compile vào trong header.



1. Thêm .css

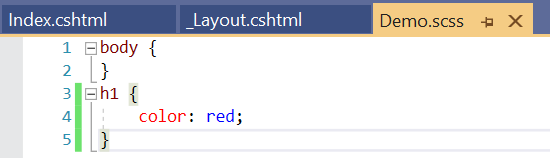
* Sau đó ta code SASS như bình thường.

File Index.cshtml



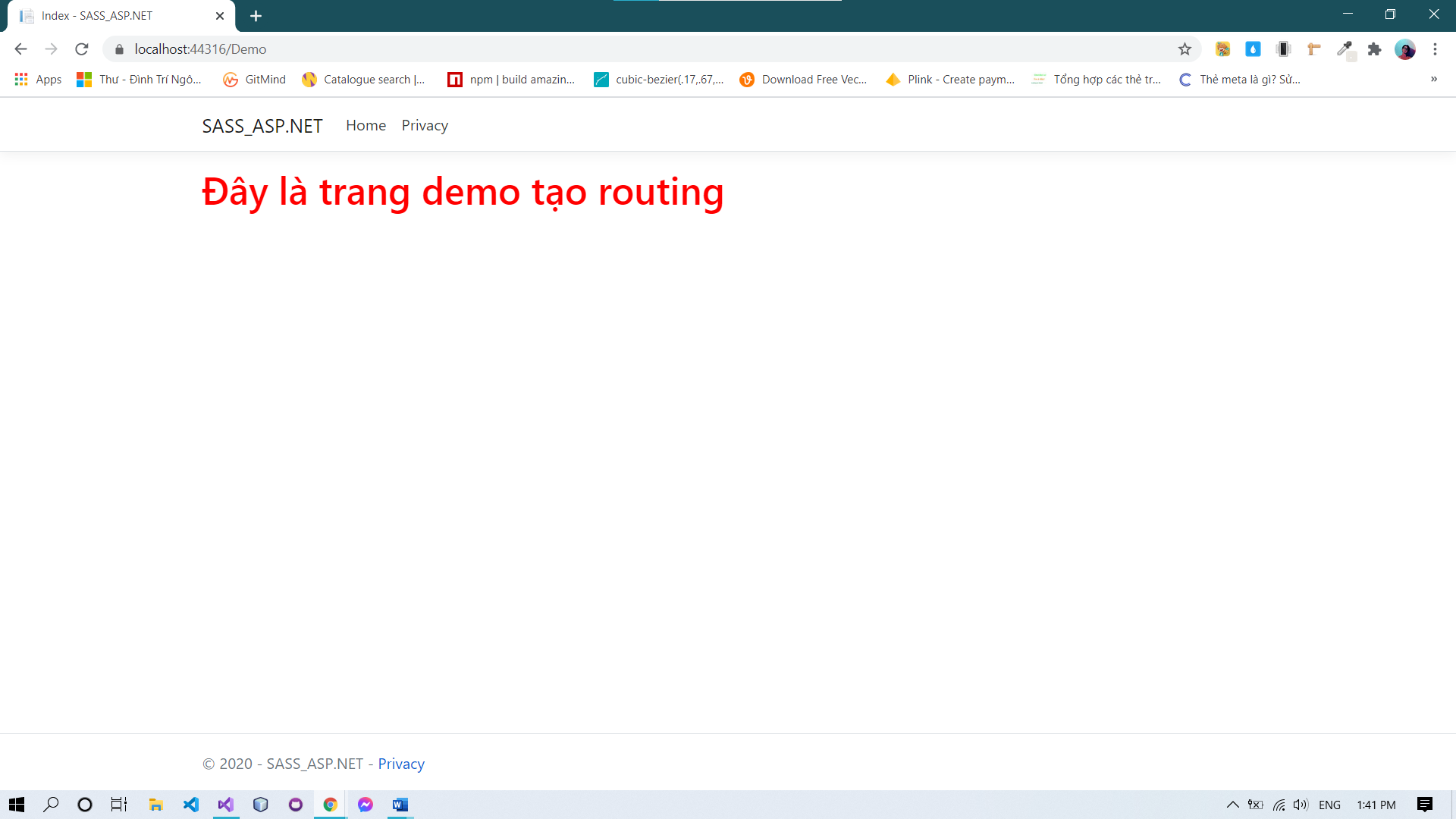
1. Index.cshtml

File SCSS



1. File SASS

Kết quả:



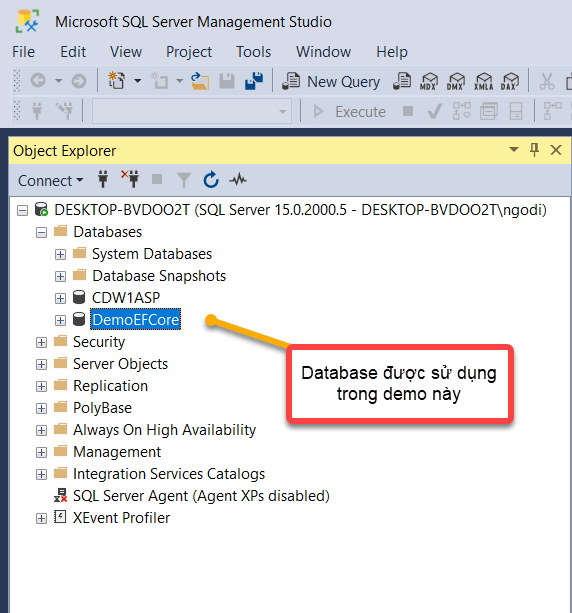
1. Kết quả đạt được

# Khái niệm căn bản trong ASP.NET Core

## Giới thiệu về EntityFramework Core

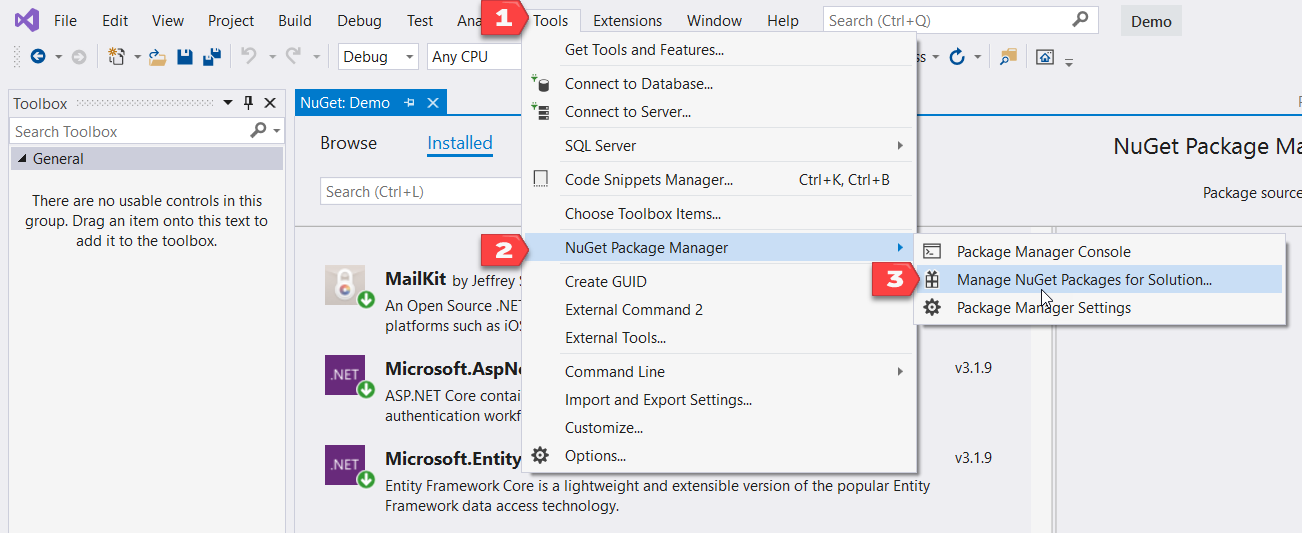
EF(Core) là một library giúp chúng ta có thể kết nối với cơ sở dữ liệu (SQL server).Với EF(Core) truy cập dữ liệu được thực hiện bằng cách sử dụng model. Model được tạo thành các class giúp ánh xạ và truy xuất hoặc nhập dữ liệu vào trong database.

**Sau đây là các cách kết nối database bằng EF(Core)**



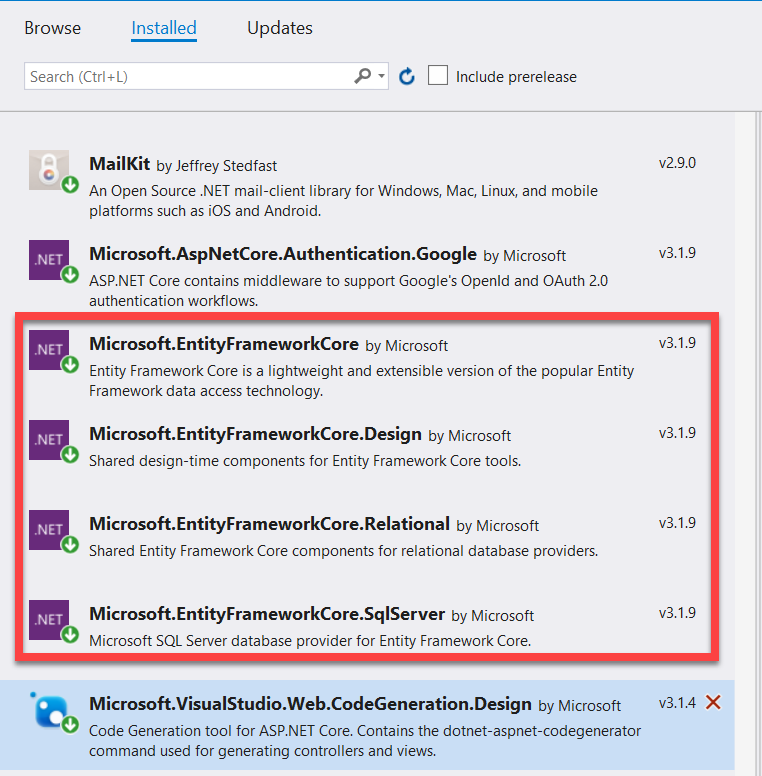
1. Database SQL Demo

Trước tiên bạn cần cài các NuGet Package hỗ trợ EF(Core). Click Tools 🡪 NuGet Package Manager 🡪 Manage NuGet Packages for Solution…



1. Các thao tác truy cập Manage NuGet Packages for Solution…

Sau đó cài các NuGet Package sau:



1. Các NuGet Package hỗ trợ EF(Core)

**B1:** Mở project của bạn sau đó truy cập vào file appsetting.json



1. Cấu hình kết nối với Database

* Khởi tạo chuỗi json như trên
* Server = “something”

Truy cập vào cơ sở dữ liệu SQL Server và lấy địa chỉ của database

* Database = “something”

Tên database cần phải truy cập

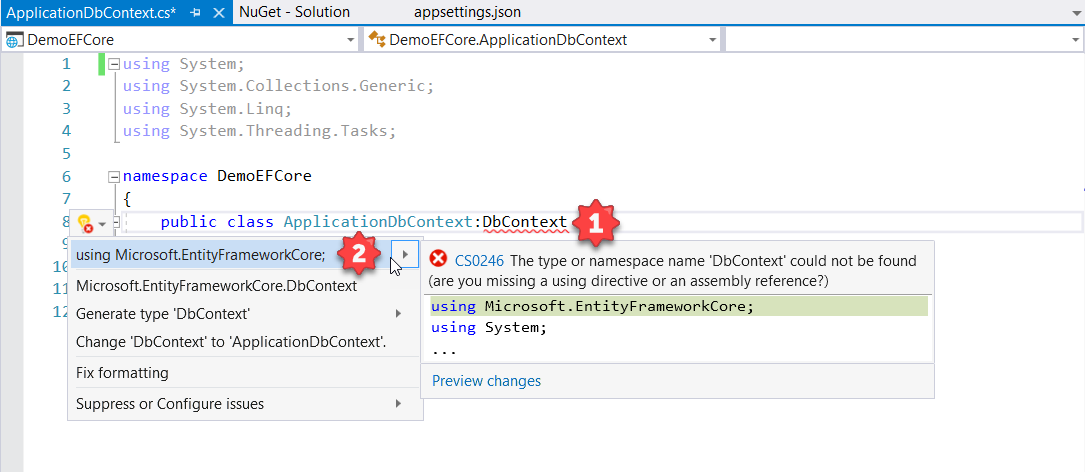
* Integrated Security = true

Mở tính năng bảo mật cho database

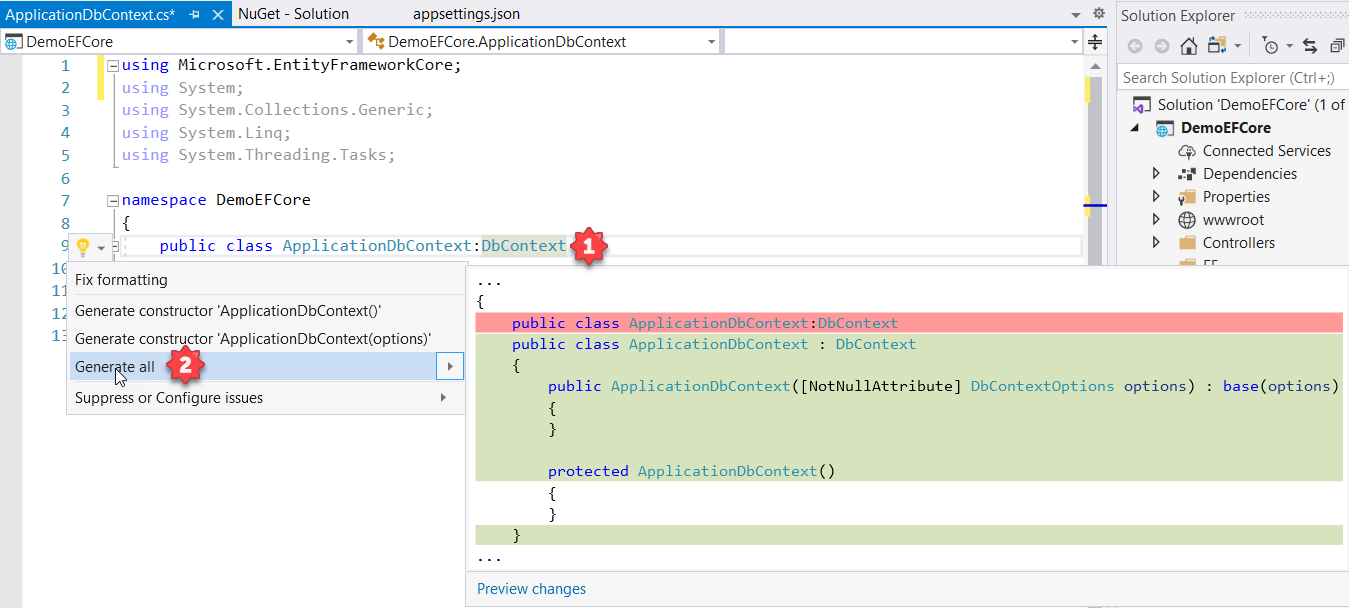
**B2:** Tạo ra một class tên là ApplicationDbContext dùng để ánh xạ Database thông qua Dbcontext.



1. Tạo một class ApplicationDbContext



1. Kế thừa DbContext. Sau đó nhấn Alt + Enter để sử dụng thư viện



1. Tiếp tục trỏ vào DbContext. Rồi Alt + Enter 🡪 Generate all. Tạo constructors.

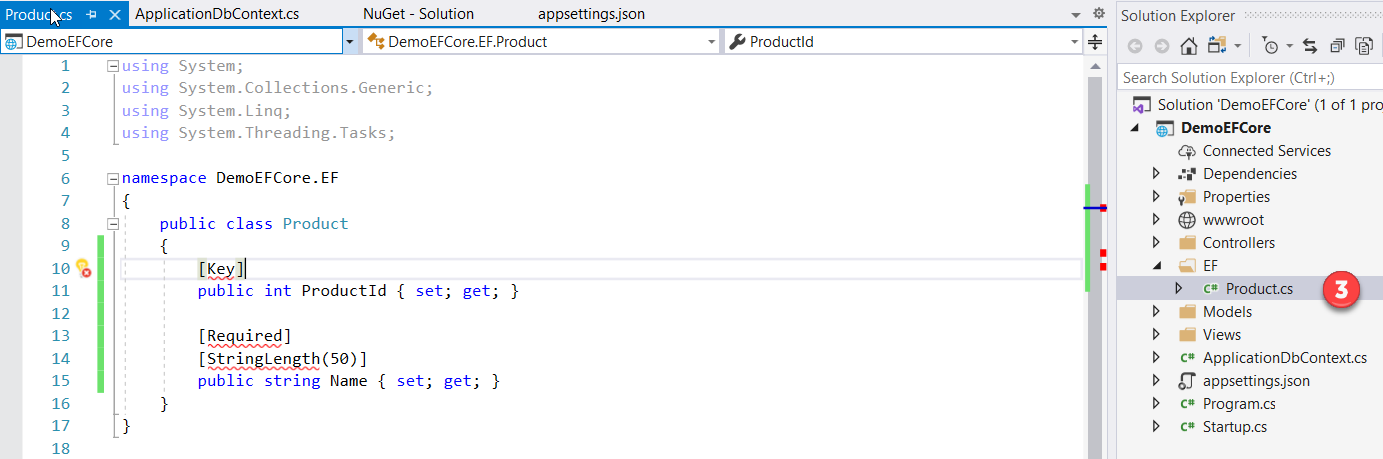




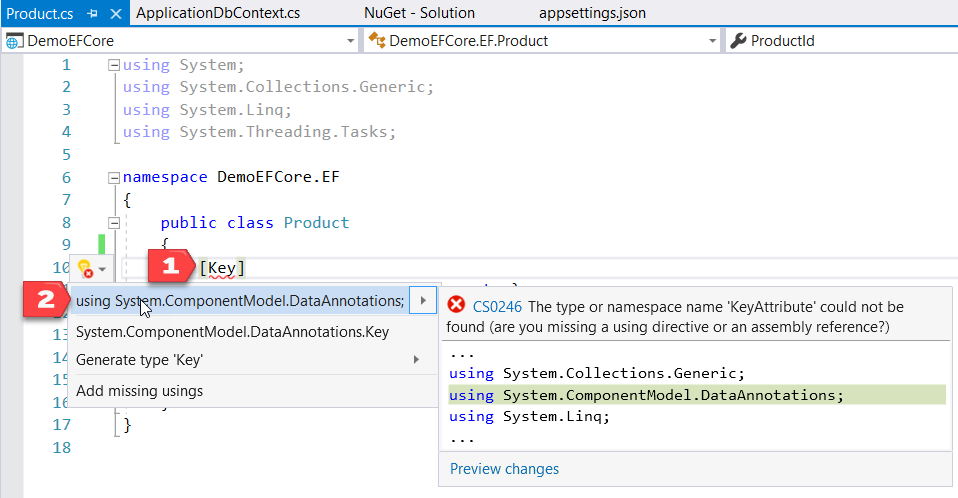
1. DbContextOptions<ApplicationDbContext> options

**B3:** Tạo một đối tượng để ánh xạ, ở đây ta tạo Product

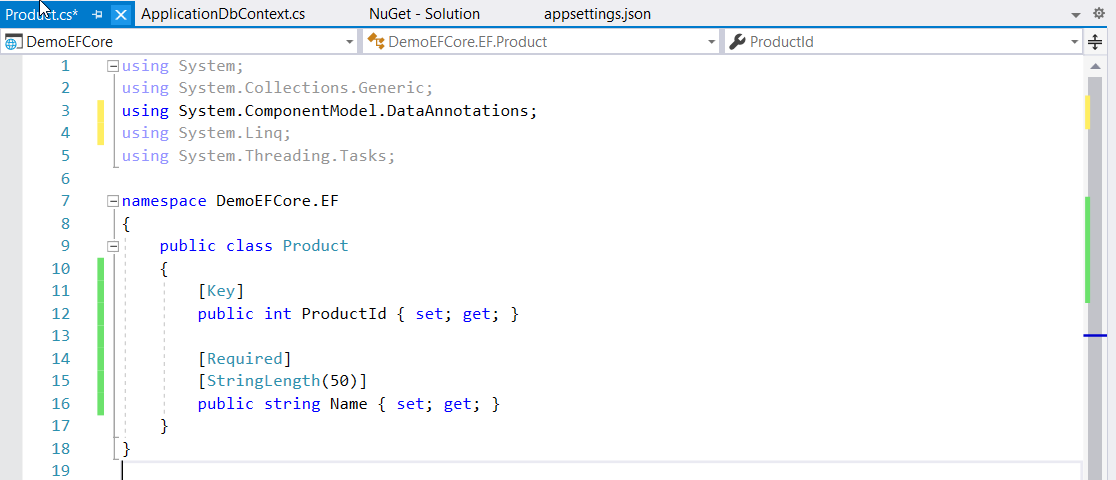
Có các primarykey ProductId và Name.



1. Báo lỗi cần thêm thư viện.



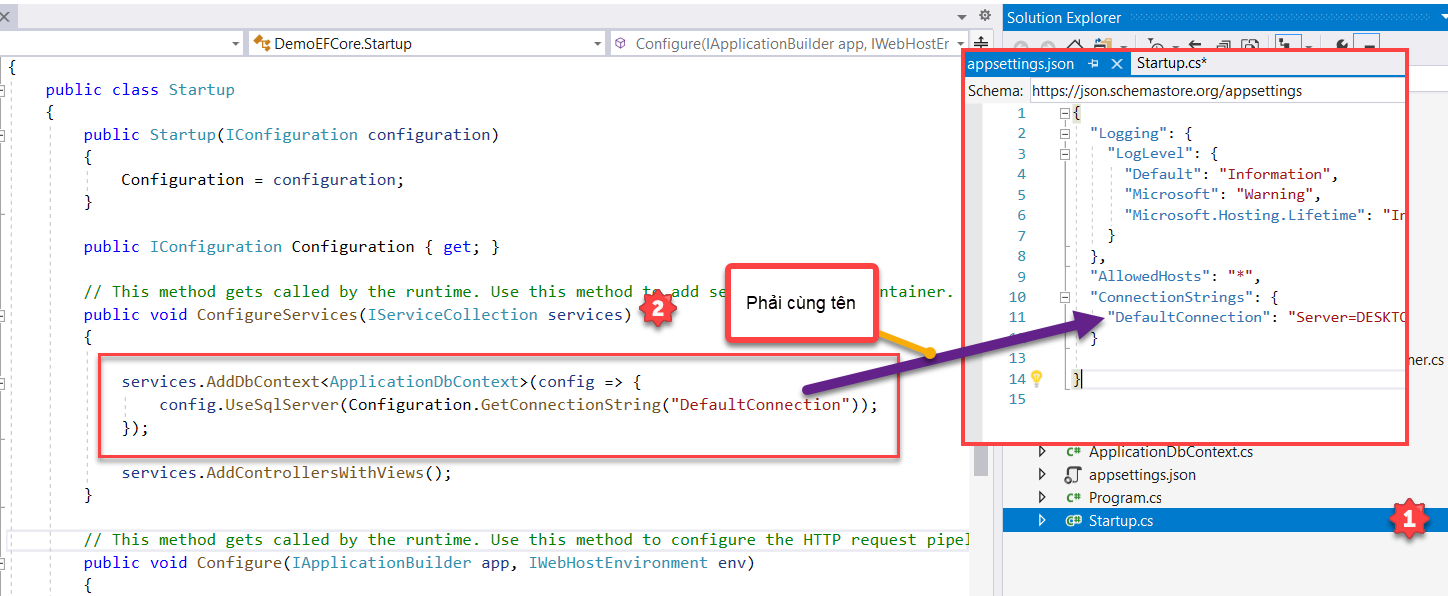
1. Trỏ vào [Key], Alt+Enter sau đó thêm thư viện vào.



1. Đối tượng Product



1. Vào lại ApplicationDbContext ánh xạ Product mới tạo vào CSDL

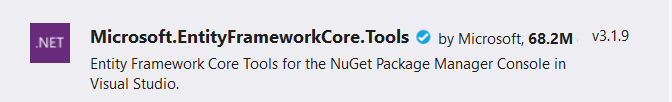


1. Kết nối Database thông qua appsettings.json

## Giới thiệu về Migration

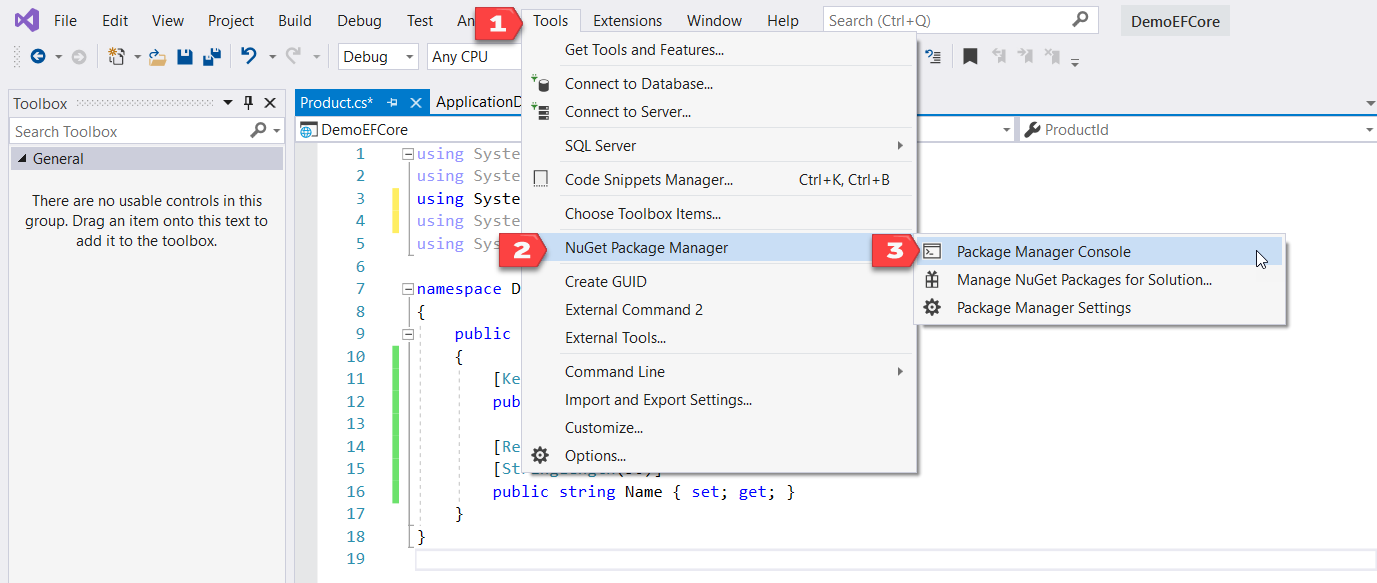
Migration là cơ chế đặc biệt trong Entity ,cho phép cập nhật các thay đổi về cấu trúc CSDL ở code lên CSDL phía server. Các phiên bản thay đổi được lưu lại, đảm bảo dữ liệu đang tồn tại không bị mất.

Để bắt đầu về Migration hãy cài đặt NuGet Package:



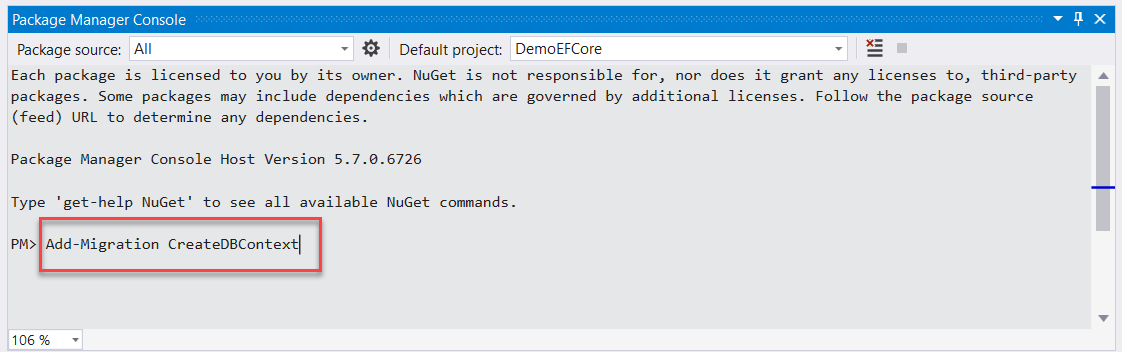
1. Microsoft.EnityFrameworkCore.Tools

Sau khi thực hiện các bước trên Entity(Core) ở trên hãy mở Package Manager Console như sau:

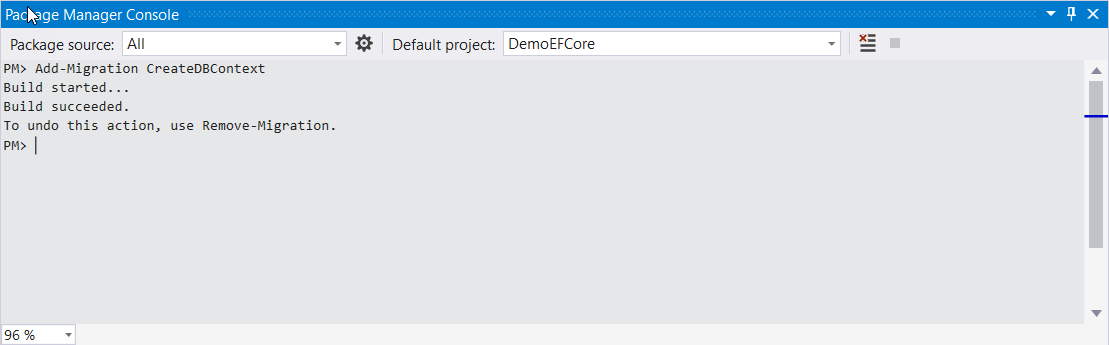


1. Tools 🡪 NuGet Package Manager 🡪 Package Manager Console

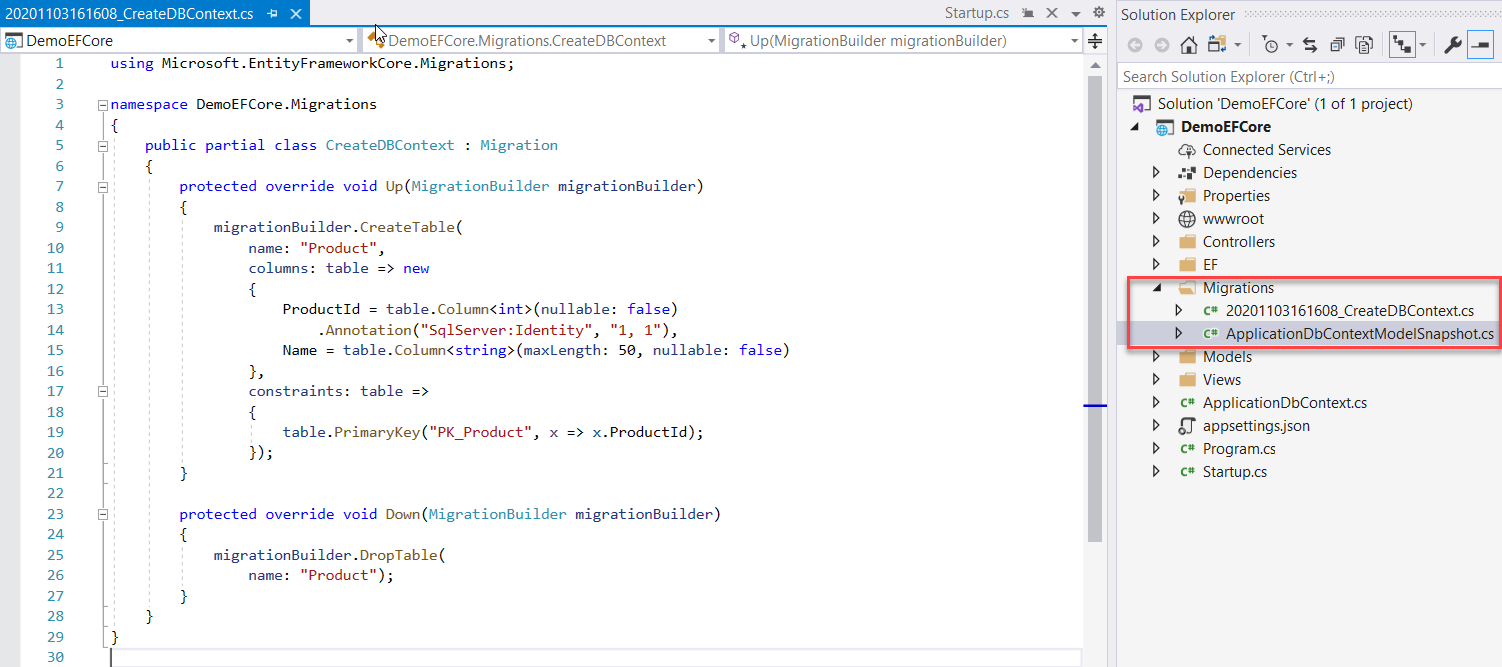
Sau đó nhập: Add-Migration CreateDBContext



1. Thêm Migration.



1. Thông báo tạo Migration thành công



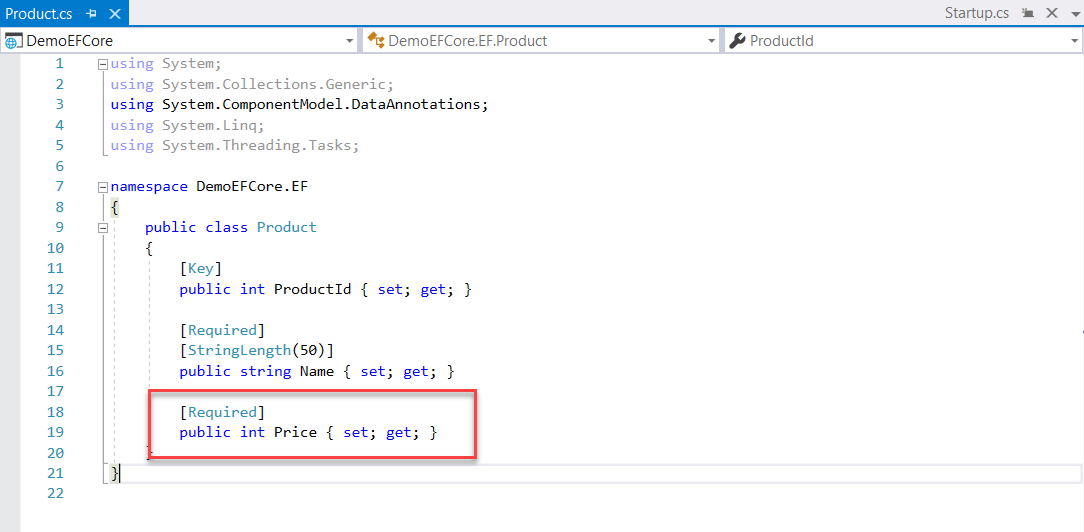
1. Migration “CreateDBContext” được tạo

“CreateDBContext” là tên của migration vừa tạo. Add-Migration giống như git commit. Lưu phiên bản hiện tại Database ở code.

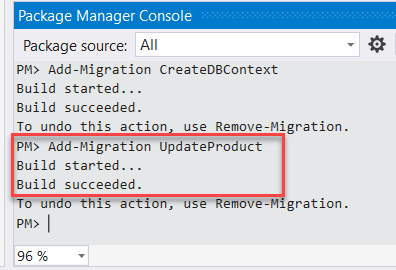
**Cập nhật Migration:**

Bạn có class Product có các properties là ProductId, Name

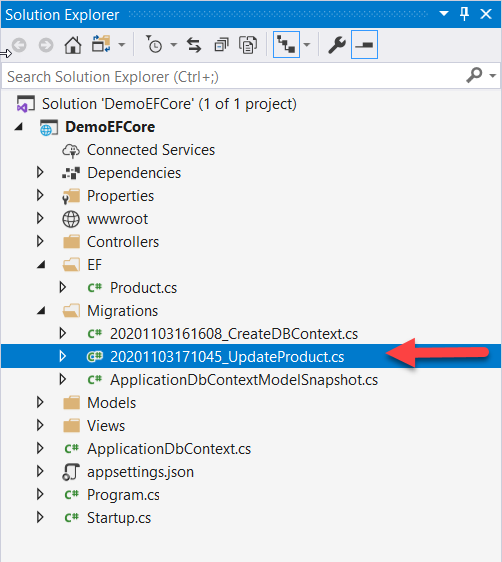
Sau đó bạn thêm Price vào trong Product. Khi thêm xong để update lên Database ta dùng lại lệnh Add-Migration và đặt tên cho Migration mới.



1. Thêm thuộc tính Price vào đối tượng Product

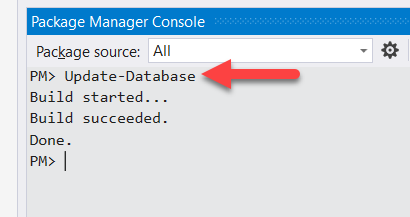


1. Cập nhật thuộc tính Price của đối tượng Product

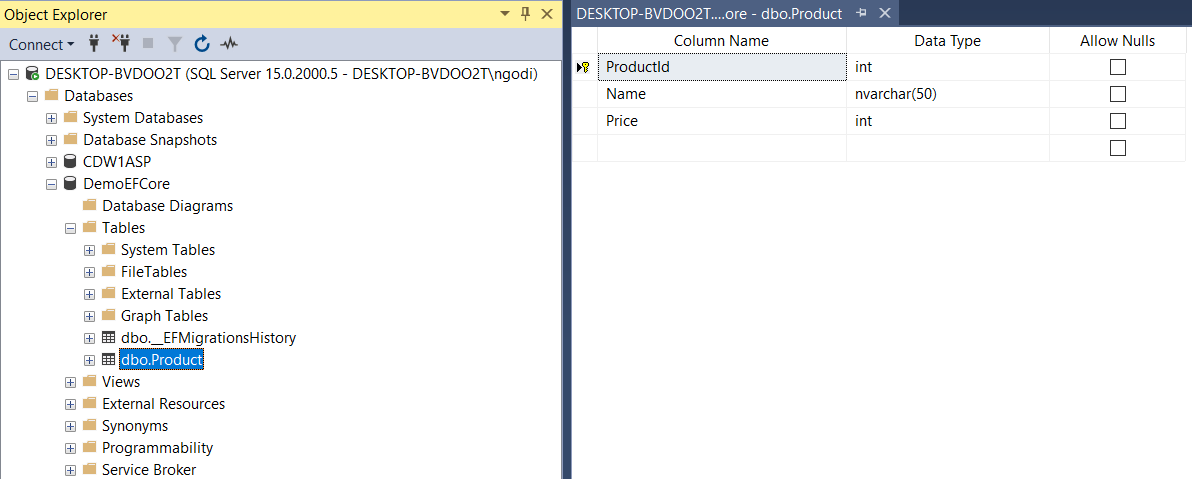


1. Migration UpdateProduct được tạo

Dùng lệnh Update-Database để đưa lên SQL trong Package Manager Console.



1. Cập nhật thành công lên SQL



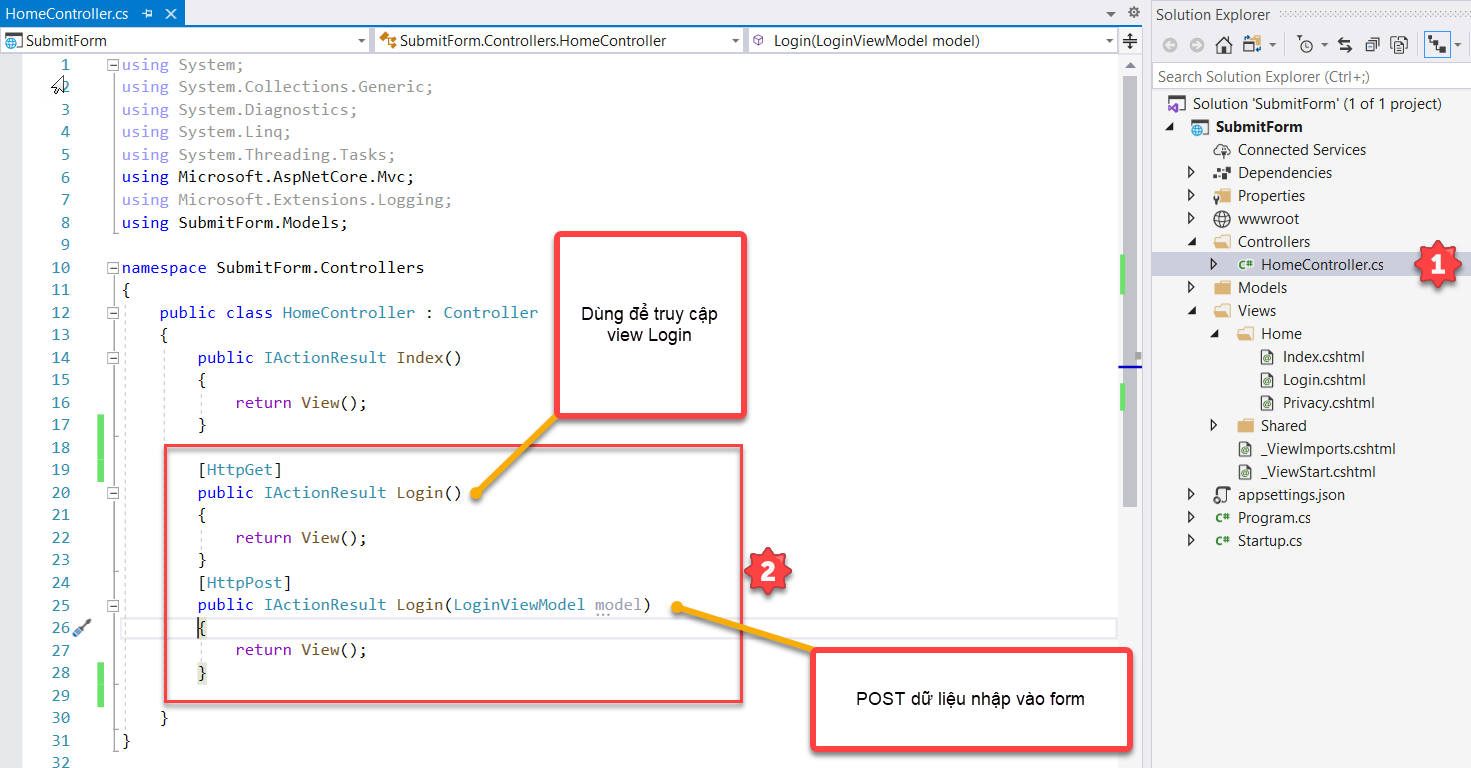
1. Dữ liệu phía SQL được tạo thành công

## Giới thiệu về Submit form

Submit form: Khi bạn muốn client gửi lên các dữ liệu của họ lên server, chúng ta dùng thẻ form trên view để đưa về cho Controller

Để Submit form ta hãy làm như sau:

**B1:** Tạo Action GET, POST cho view Login.



1. Login Controllers.

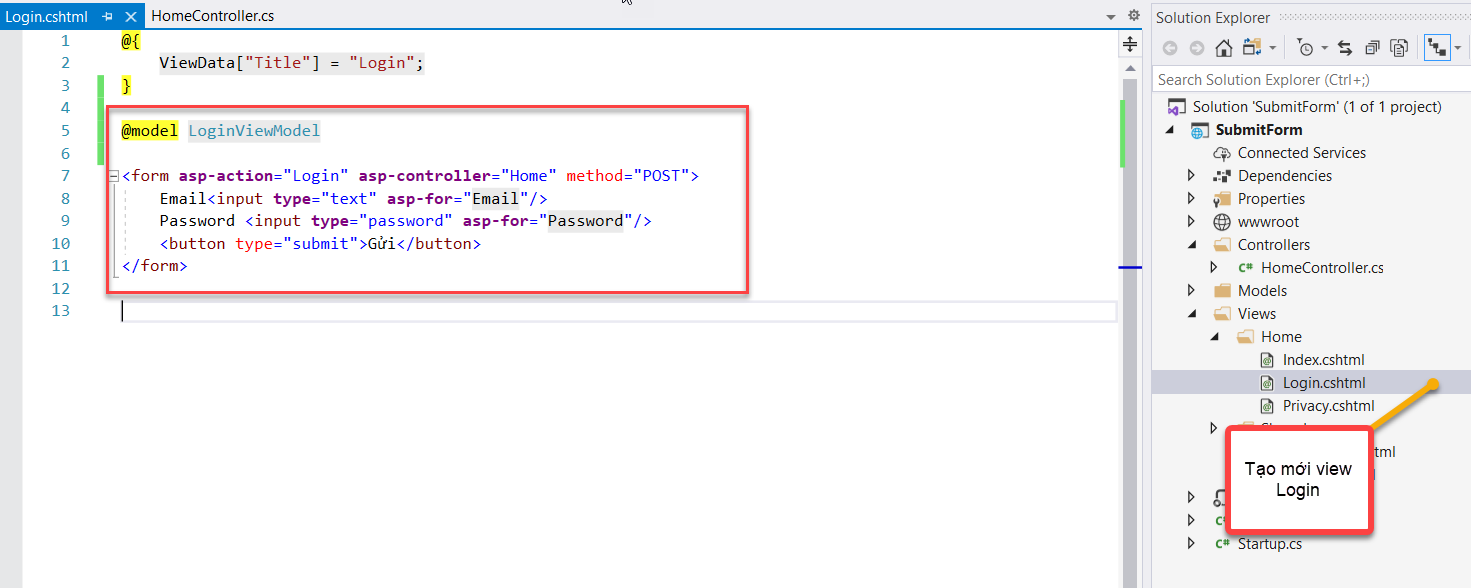
* Với HttpGet Được xem như URL đầu vào của Login

Method :get

* Với HttpPost được xem như URL có dữ liệu đưa vào

Method:post

Bước tiếp theo bạn hãy tạo view Login và add thông tin ở dưới vào.

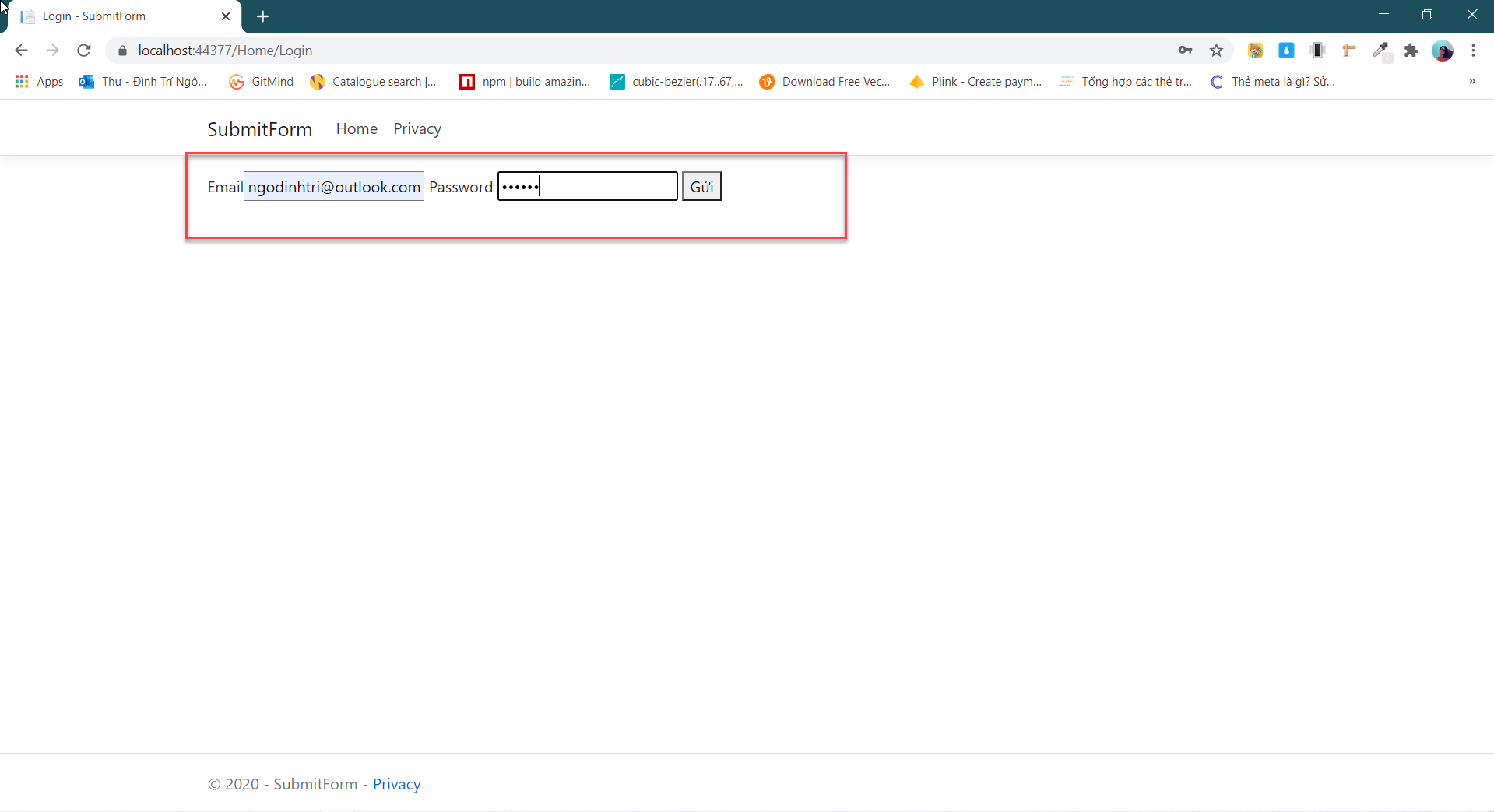


1. Tạo form trong Login.cshtml



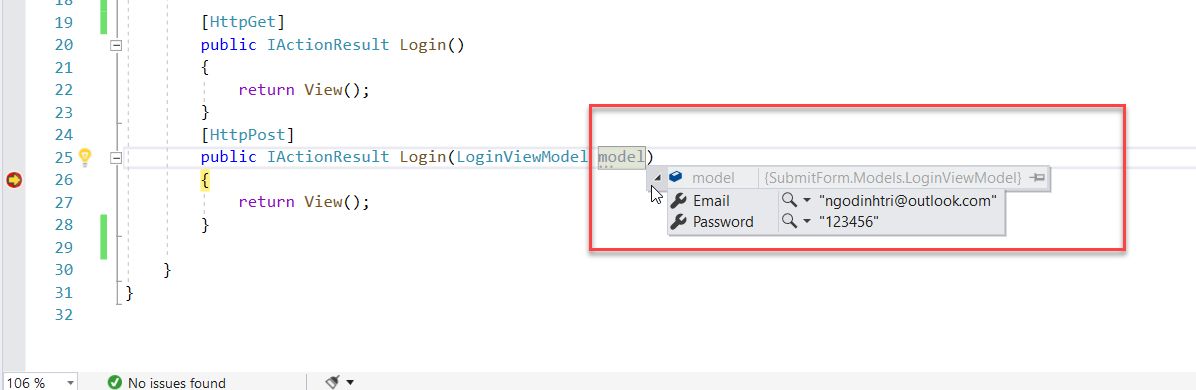
1. Đặt BreakPoint trong Login Controller

* Run project. Truy cập vào trang (localhost)/Home/Login. Nhập dữ liệu, rồi gửi.



1. Trang login sau khi chạy Project.

* Ta được kết quả như sau



1. Kết quả khi gửi đi.

## Giới thiệu về truy vấn dữ liệu bằng LINQ

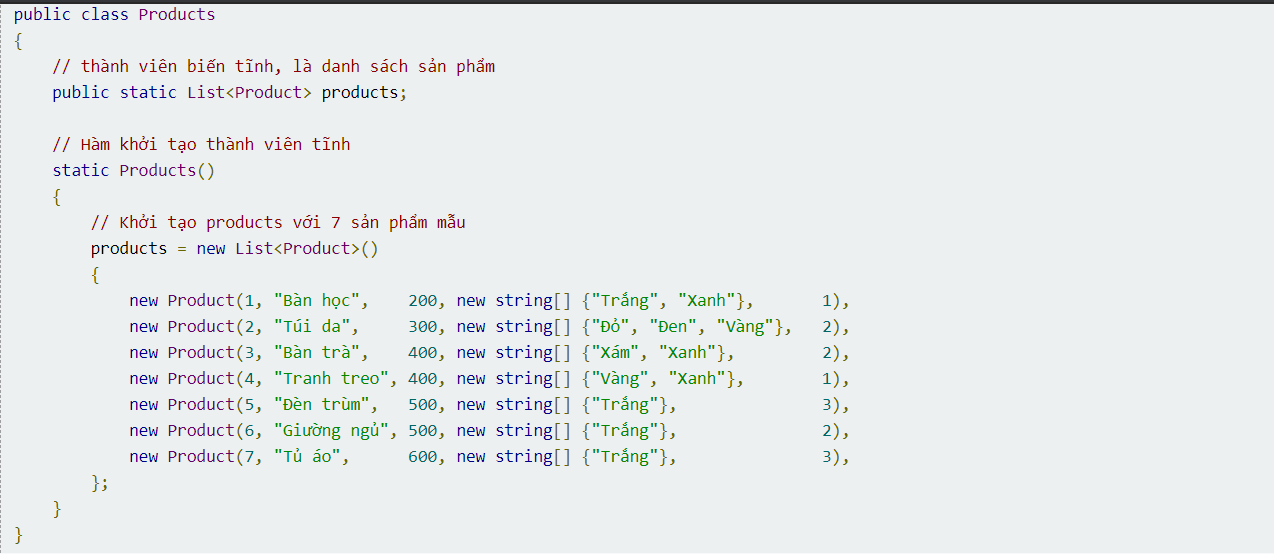
Ngôn ngữ truy vấn LINQ cho phép truy vấn,chuyển truy vấn thành các câu lệnh trên đối tượng để truy vấn vào Database

* VD truy vấn LINQ:

Tạo một đối tượng tên là Product



1. Đối tượng product viết bằng LINQ



1. Thêm sản phẩm



1. Demo

Tham khảo tại:

<https://docs.microsoft.com/en-us/dotnet/api/system.linq?view=netcore-3.1>

## Giới thiệu về Identity Authentication

Identity Authentication là là model (built-in) nó cung cấp cho người dùng các tính năng đầy đủ và đa dạng về authentication và authorization, cho phép tạo tài khoản, đăng xuất cũng như là thêm, xóa user... Hỗ trợ tích hợp với external login bên ngoài như Google, Facebook, Twitter... Và được lưu vào hệ sinh thái.

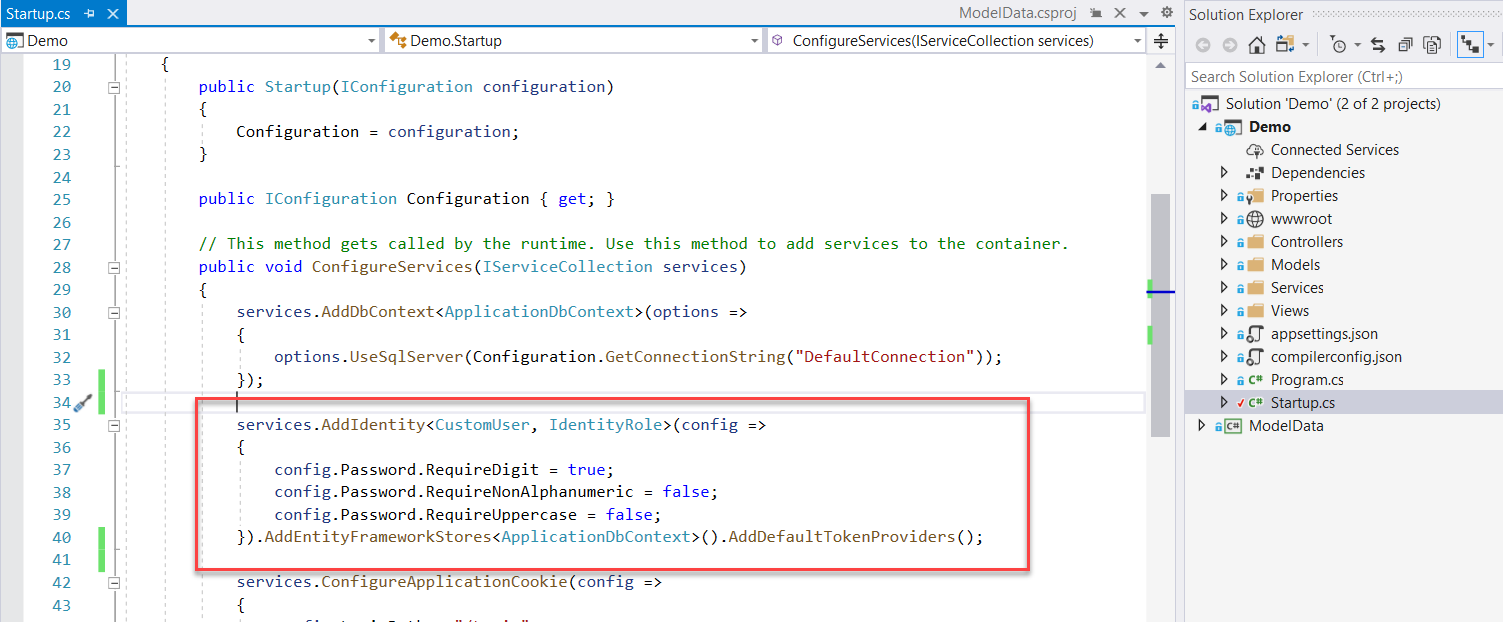
## Hướng dẫn thiết lập Identity

B1: Cài Identity NuGet Package.



1. Microsoft.AspNetCore.Identity.EntityFrameworkCore

B2: Vào startup.css



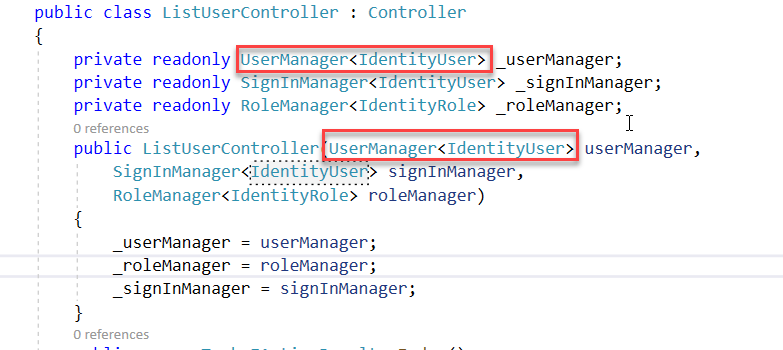
1. Custom mật khẩu cho phía identity.

* Khi này bạn có thể sử dụng identity authentication, ở config => Đây là nơi bạn có thể custom mật khẩu cho phía Identity.

## Giới thiệu về UserIdentity và User Manager

* UserIdentity: Dùng để lưu trữ thông tin người dùng
* UserManager: Dùng để khởi tạo người dùng trong database, Hãy coi một số ví dụ sau

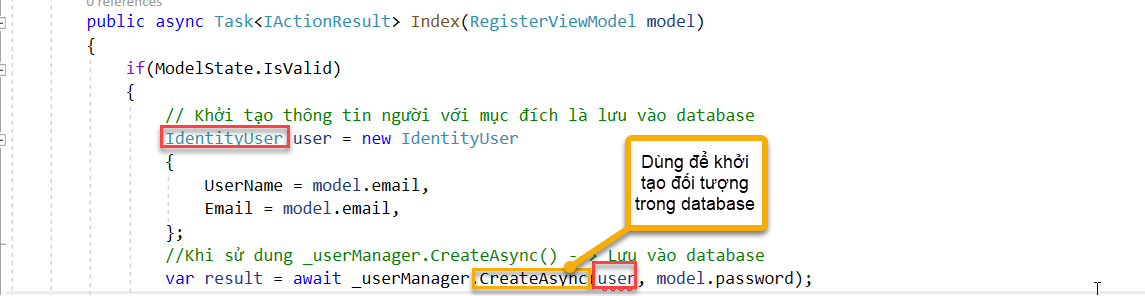
**B1:** Khởi tạo Controller mong muốn



1. Khởi tạo đối tượng quản lý người dùng và đăng nhập

* Khi này bạn có thể dùng \_userManager để thêm người dùng

**B2:**



1. Khởi tạo thông tin user trong database và đăng nhập

**\_userManager.CreateAsync ()**

* Khởi tạo một user trong Database,Khi người dùng gửi request kèm thông tin, lúc này trong database sẽ lập tức thêm vào trong ASP User table trong database

# Khái niệm nâng cao trong ASP.NET Core

## Hướng dẫn gửi email MailKit

**B1:** Download <https://www.nuget.org/packages/MailKit/> Mailkit package

**B2:**Truy cập vào [https://www.hostinger.vn/huong-dan/lam-the-nao-de-su-dung-Google-smtp-server-mien-phi/](https://www.hostinger.vn/huong-dan/lam-the-nao-de-su-dung-google-smtp-server-mien-phi/). Đây là cách đăng ký một smtp server để bạn có thể gửi email và các dịch vụ khác

**B3:**



1. Mô tả hình thức gửi message qua email

* MimeMessage => Tạo ra một đối tượng message
* From.Add() => Đây là email người gửi
* From.To()=> Đây là email người nhận
* Subject() => Tiêu đề gửi
* Body()=> Là kiểu gì, với “plain ” định dạng kiểu text văn bản
* SmtpClient()=>Khởi tạo một đối tượng smtp để gửi
* Authentication xác nhận email người gửi
* Send(đây là tin nhắn bạn gửi)
* Sau đó dùng lệnh Disconnect để kết thúc việc gửi, nếu k đóng lại sẽ gây ra lỗi

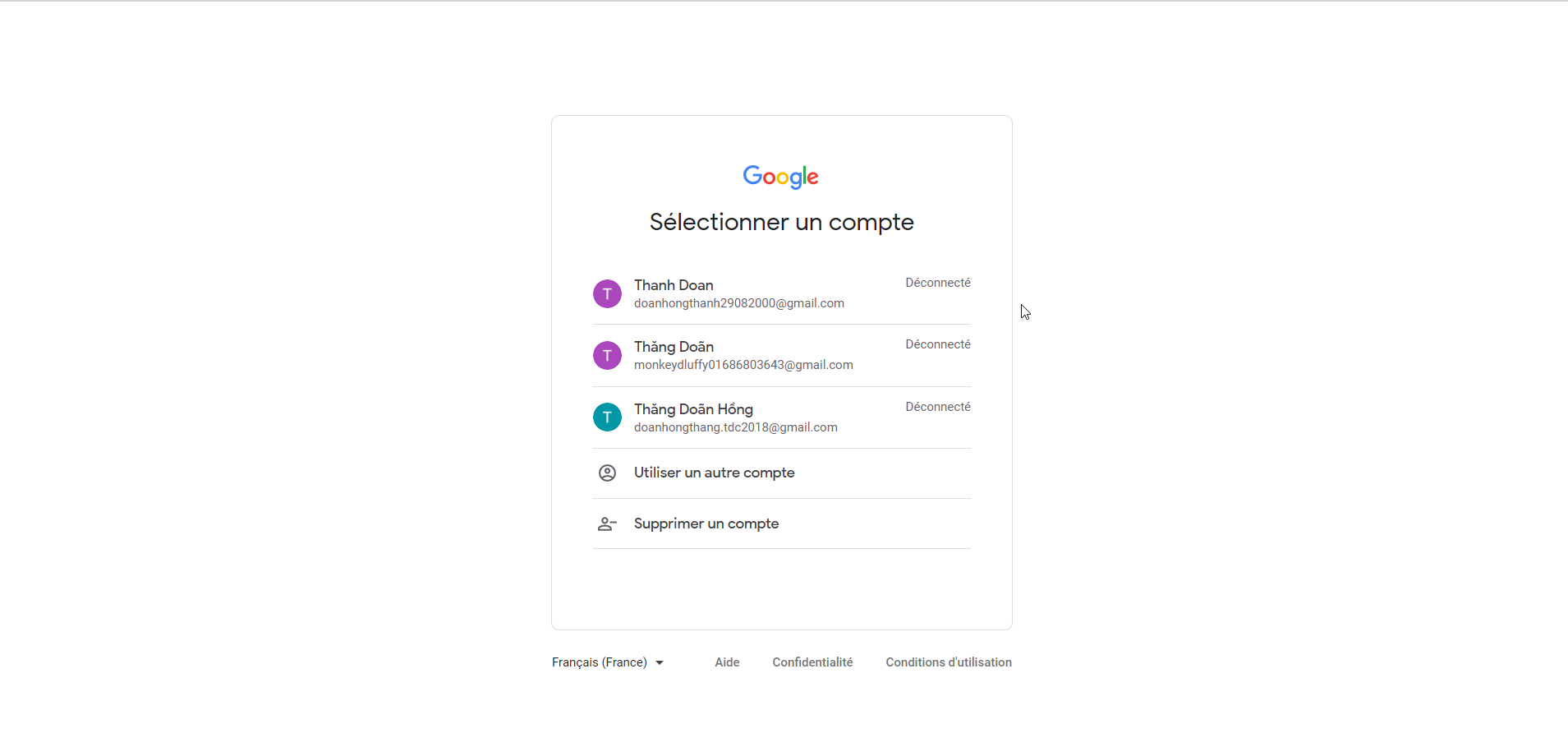
## Hướng dẫn tích hợp API login with Google

* Để tích hợp API Google login qua web application, phải có căn bản của Identity Authentication của ASP.NET Core
* Biết sử dụng signinManager methods and userManager method nhé
* Bài chỉ hướng dẫn cách tích hợp ứng dụng web của bạn với phần đăng nhập bên phía Google còn về phần ứng dụng desktop thì bạn nên nghiên cứu thêm
* Biết phát triển web bằng ASP.NET Core

**Bước 1:** Vào trang console.developer.com

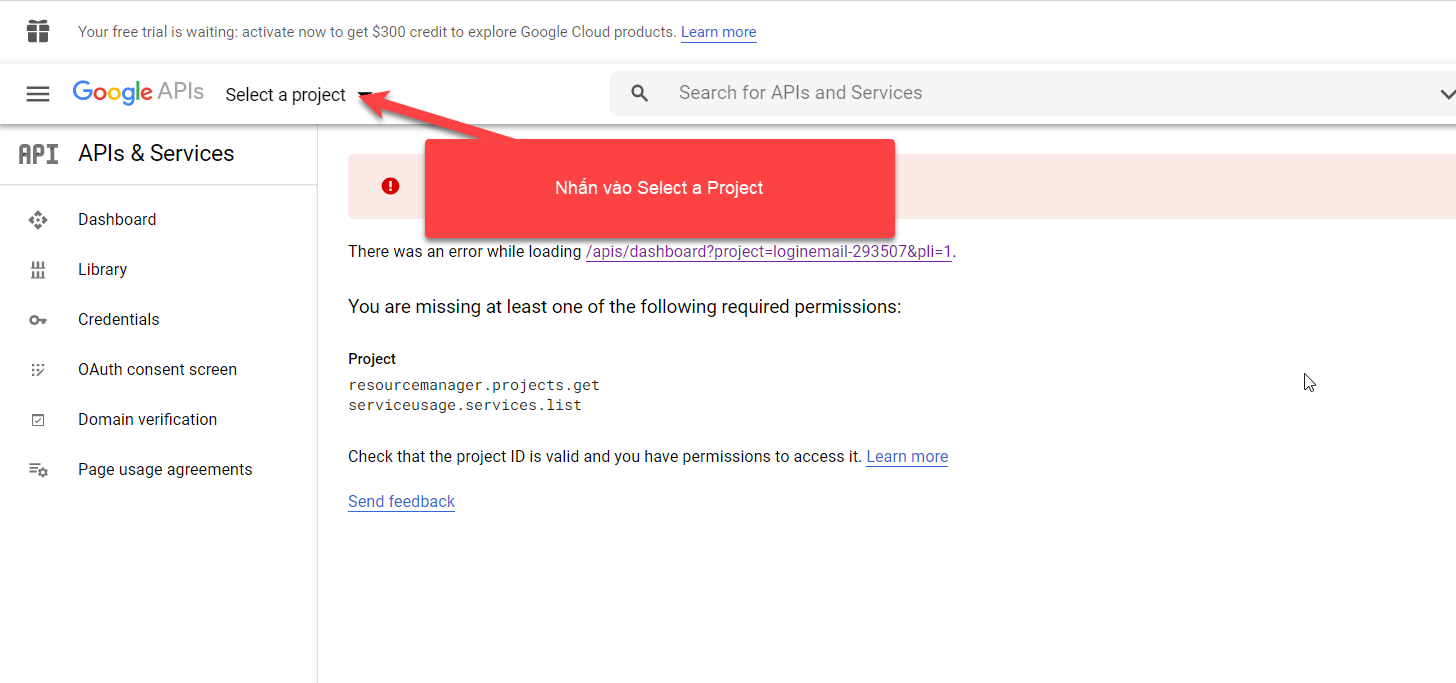
<https://console.developers.Google.com/apis/dashboard?project=loginemail-293507>

**Bước 2:** Đăng nhập vào tài khoản Google

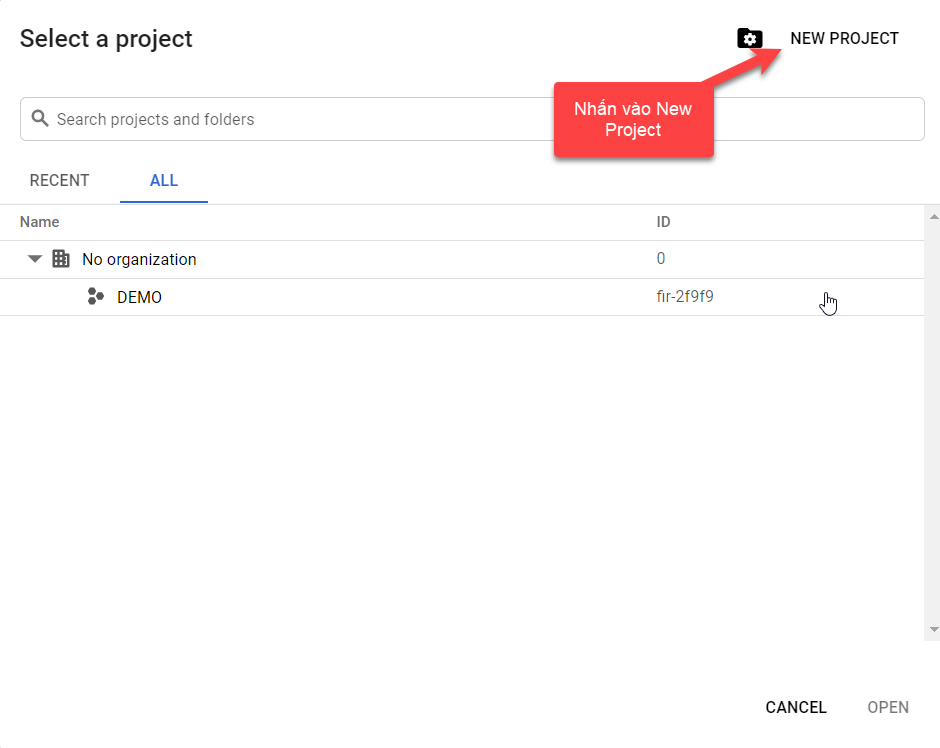


1. Đăng nhập tài khoản Google

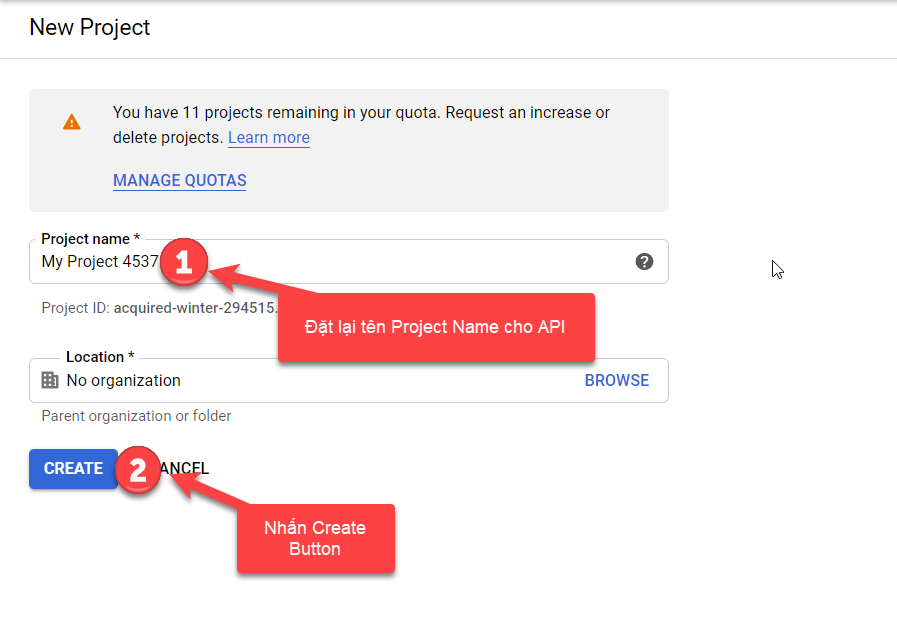
**Bước 3:** Sau khi đăng nhập hoàn tất, tạo project.



1. Lựa chọn project trong APIs Services Of Google



1. Tạo mới Project trong APIs Services Of Google

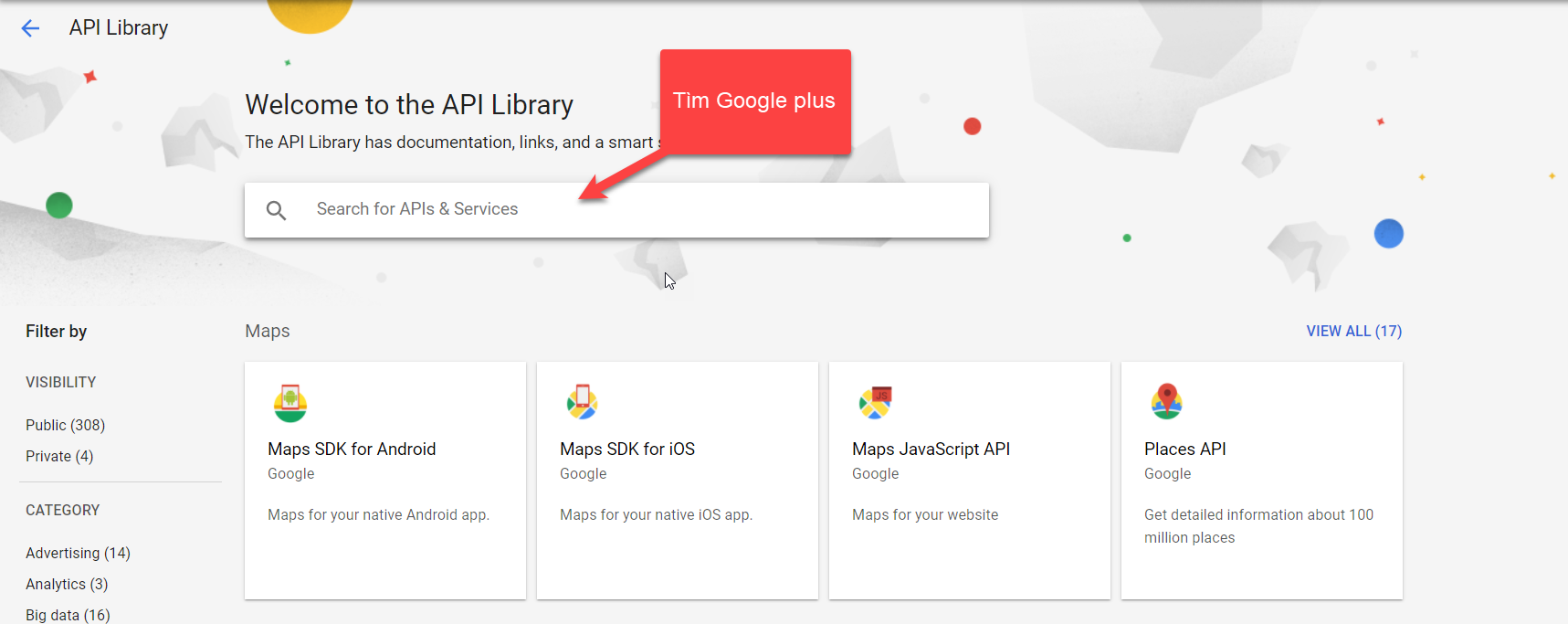


1. Đặt tên cho Project API

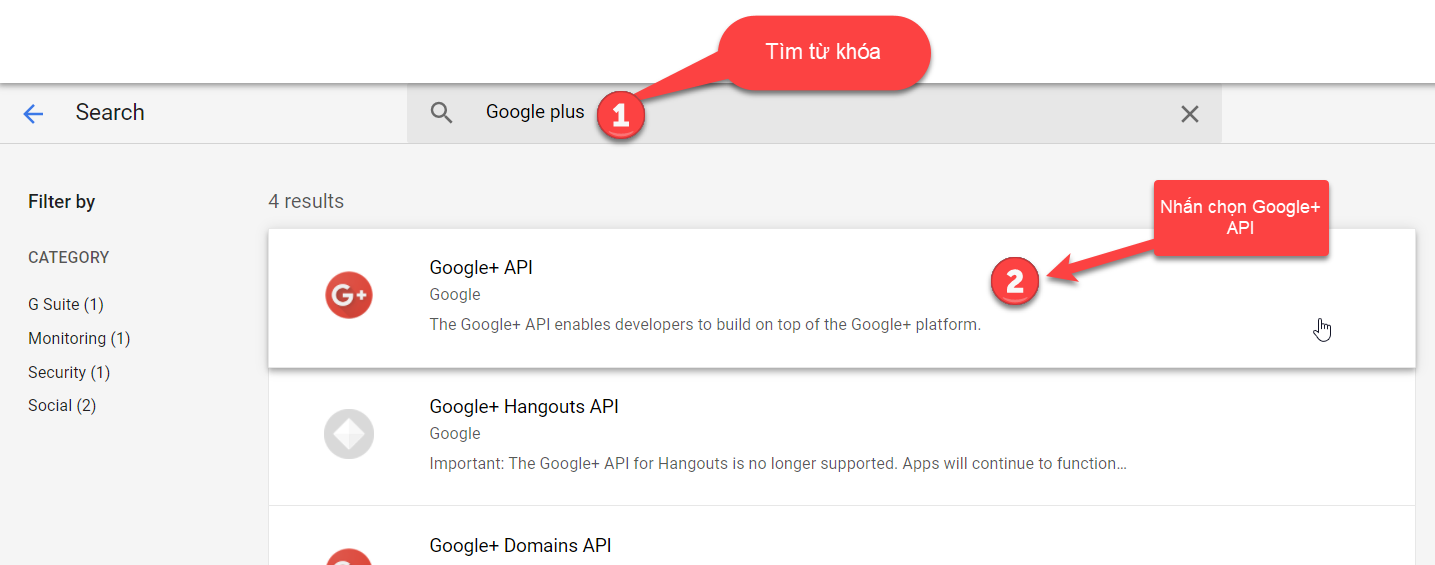
**Bước 4:** Tìm library về Google plus API



1. Mở các thư viện có trong APIs Services Google



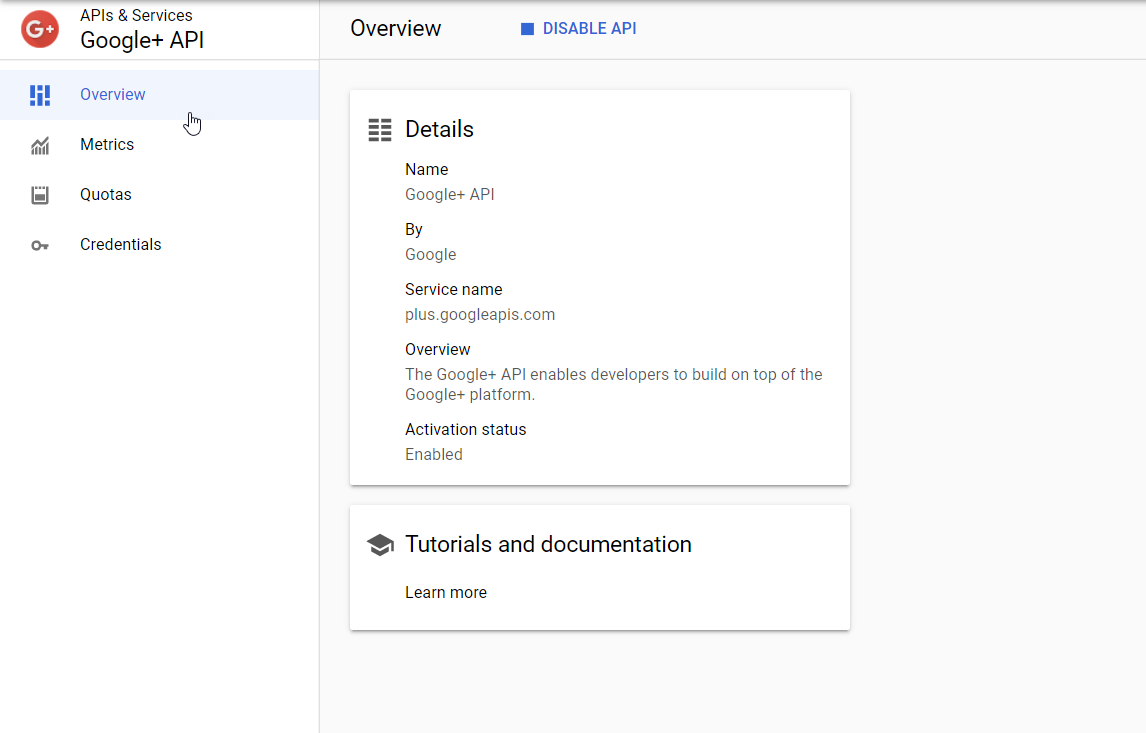
1. Tìm Google plus API



1. Tích hợp Google+ API

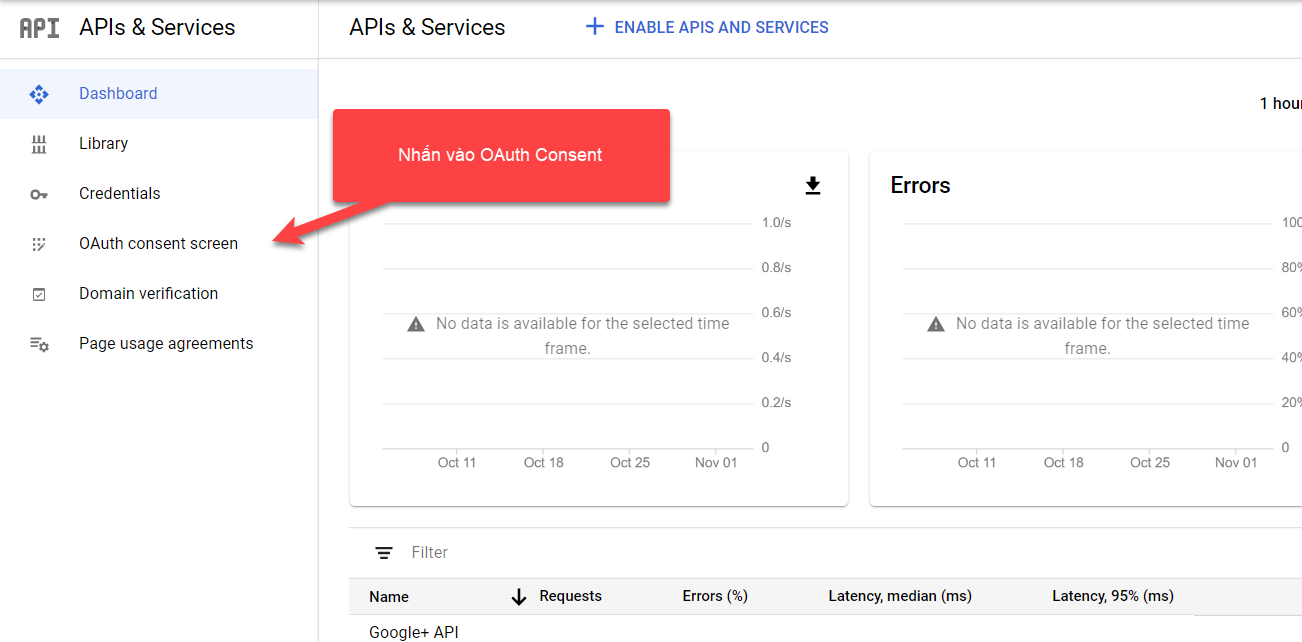


1. Kích hoạt Google+API

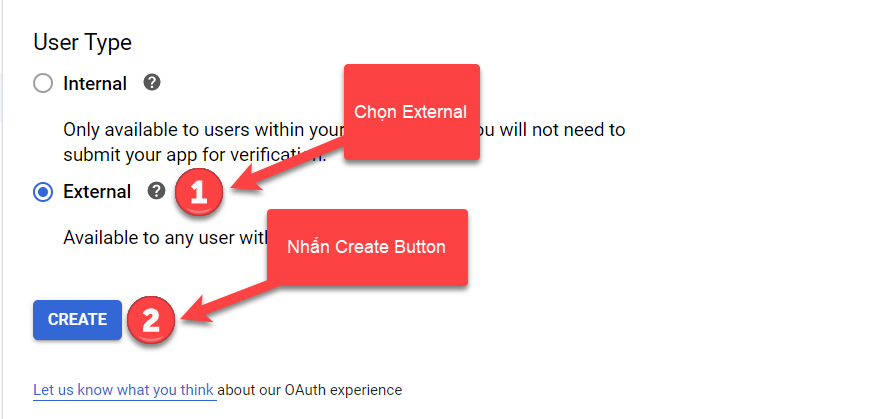


1. Hình hiển thị thành công sao khi dùng Google+ API

**Bước 5:** Khởi tạo auth client Id và Secret Id cho Người dùng



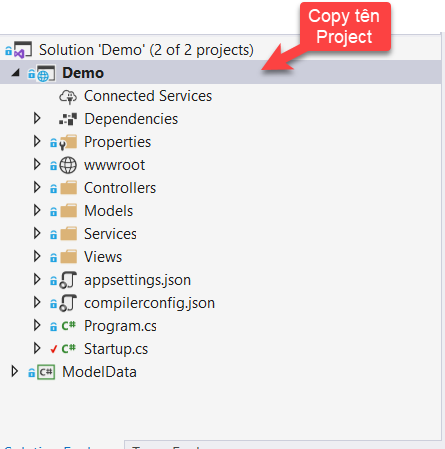
1. Khởi tạo Oauth Client Id và Secret Id



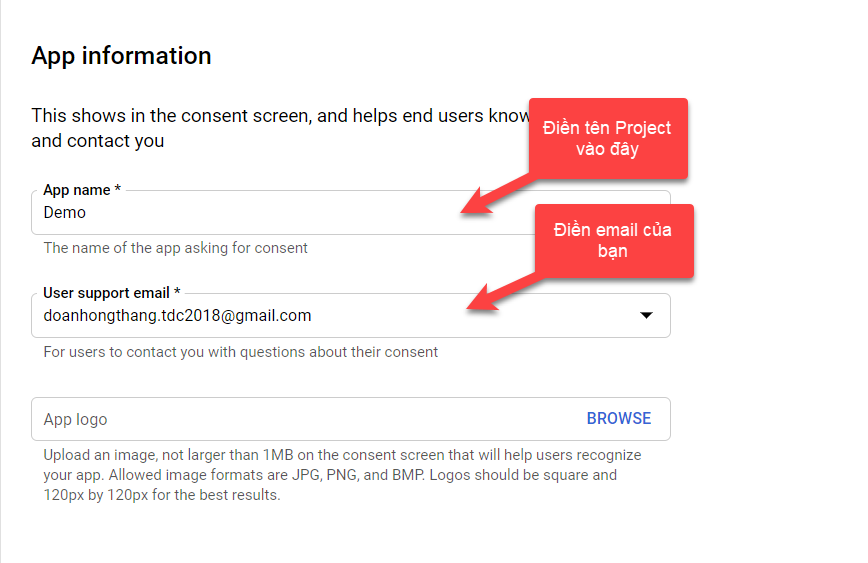
1. Xác nhận việc cho phép nhiều người dùng đăng nhập

**Bước 6:** Thiết lập tên Application API trong Services Of Google

* Lưu ý, sau khi xong Bước 5
* Mở project lên và copy tên project

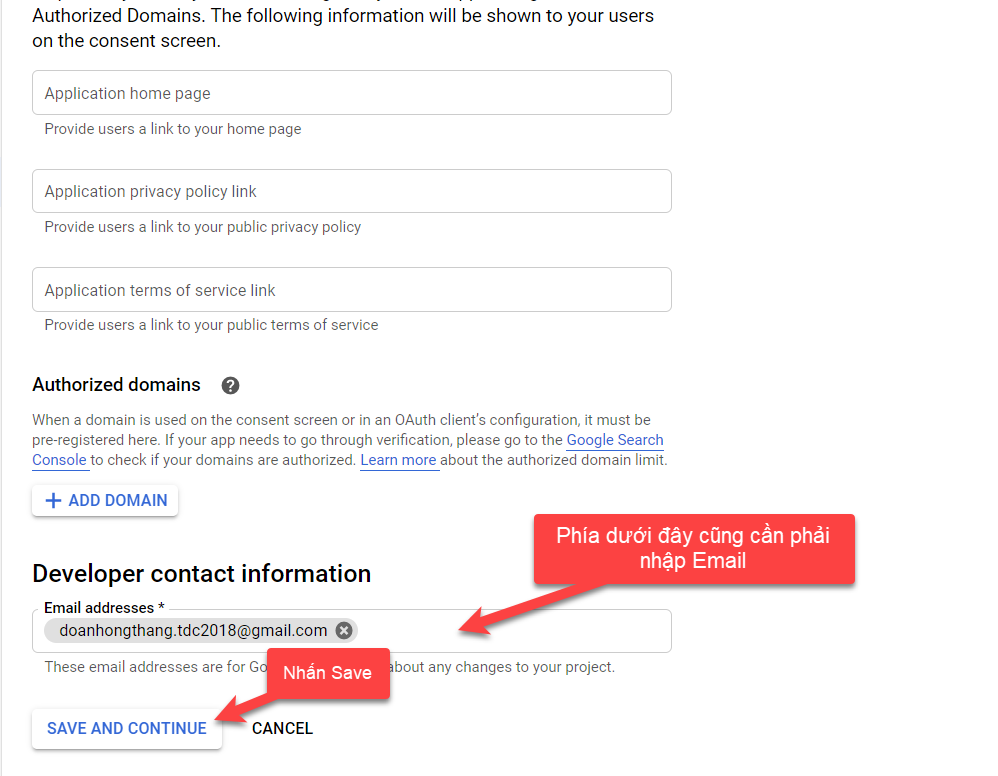


1. Copy tên project

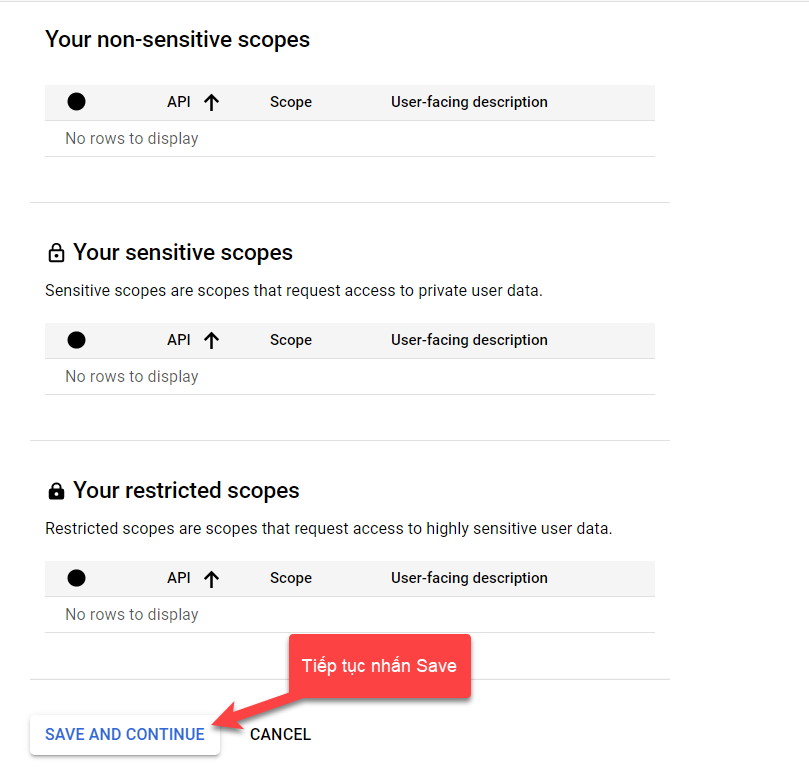


1. Điền Tên Project vào đây

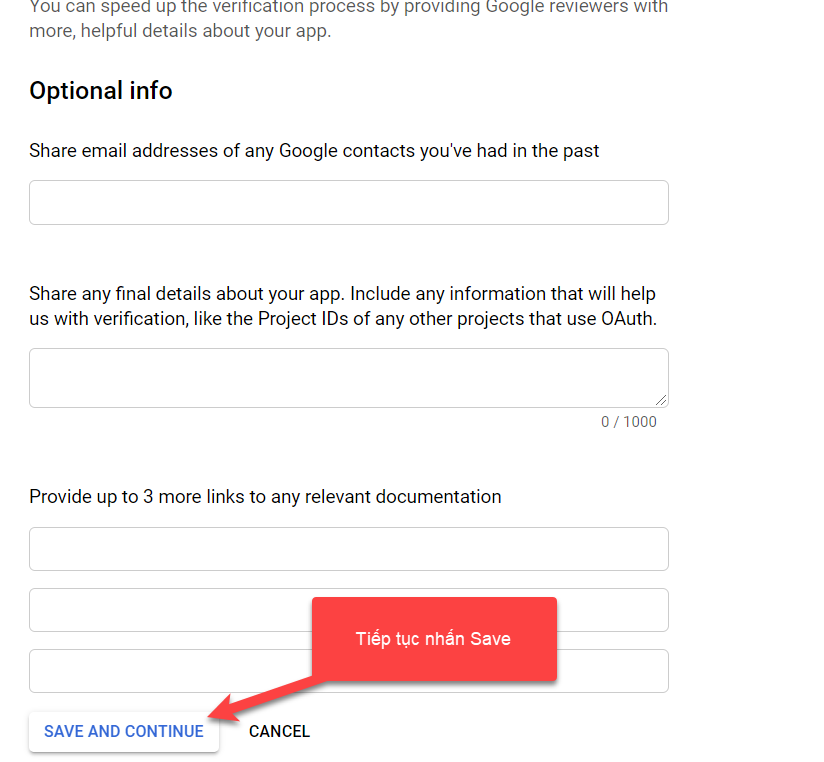
* Sau đó Kéo xuống phía dưới cùng



1. Lưu thông tin project API

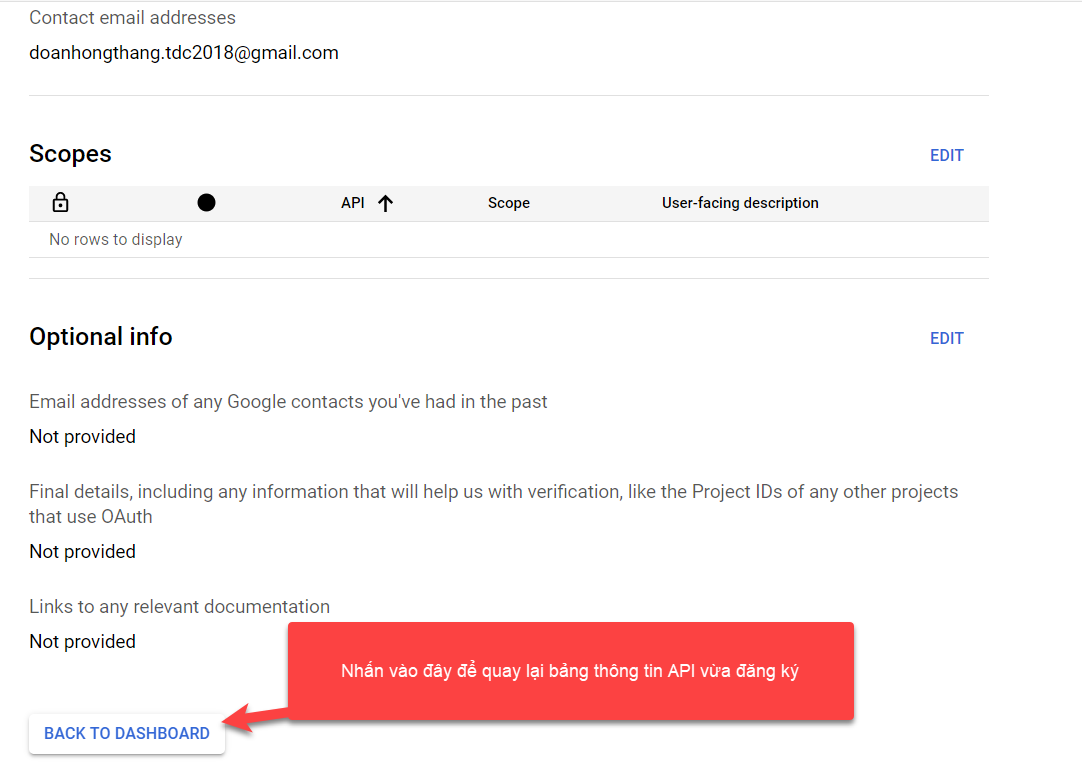


1. Scopes trong Services Of Google



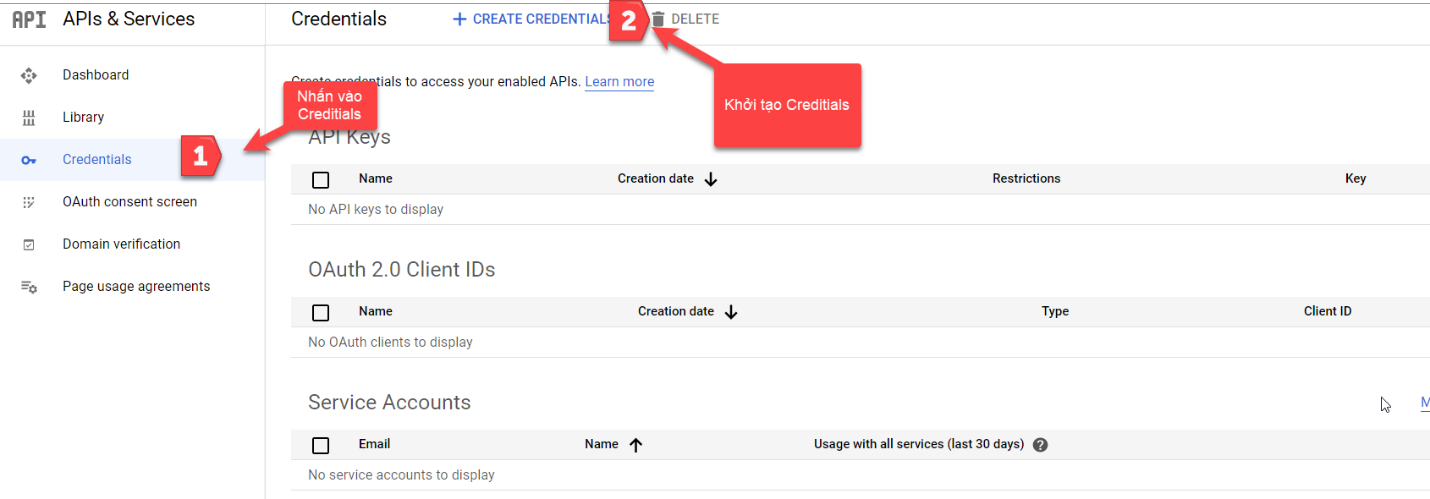
1. Thông tin về giới thiệu dịch vụ Google

* Sau khi nhấn Hoàn thành các bước trên, kéo thẳng xuống cuối trang

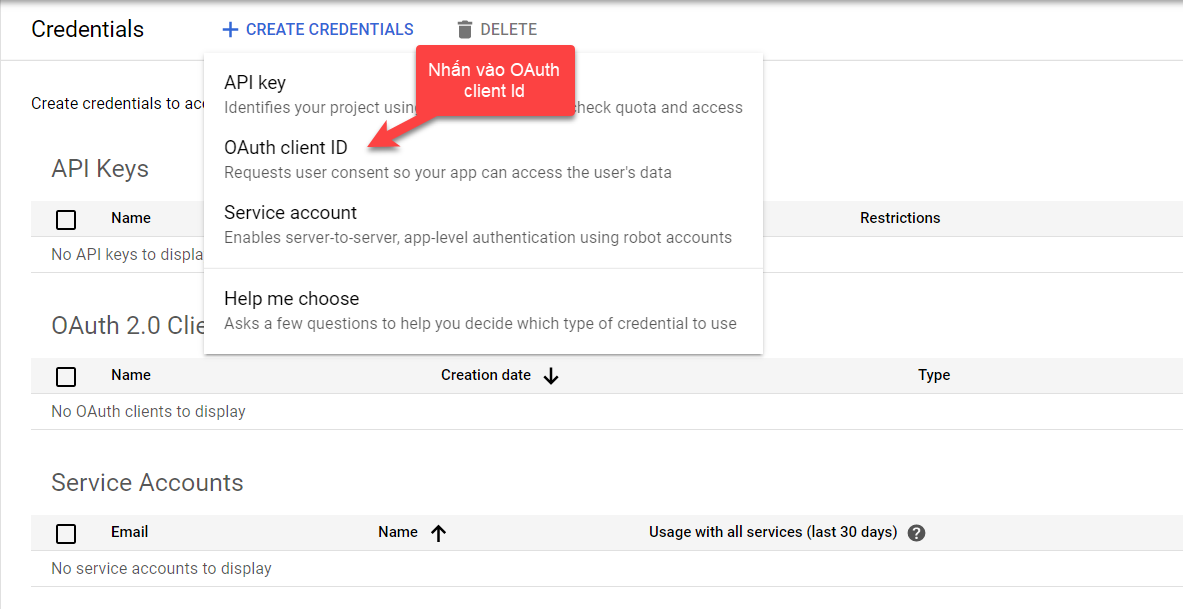


1. Quay trở lại trang thông tin API

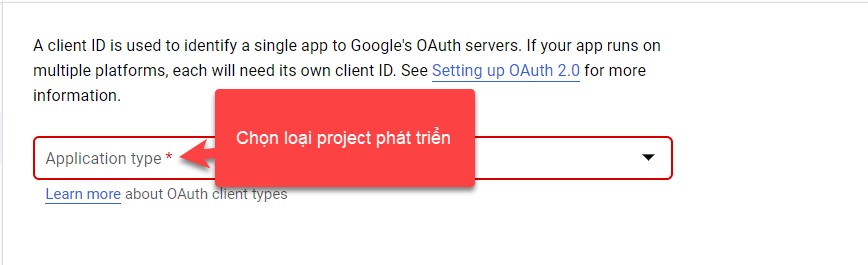
**Bước 7:** Sau khi quay lại trang chính, nhấn Credentials để khởi tạo thông tin sử dụng



1. Khởi tạo Credentials cho API

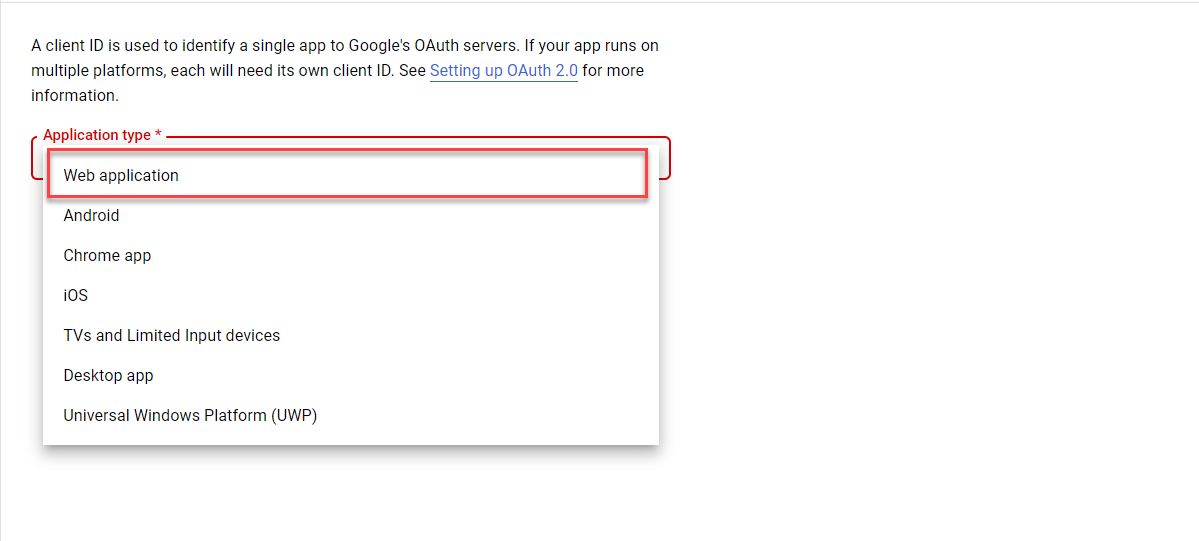


1. Khởi tạo Client Id



1. Chọn loại project đang phát triển

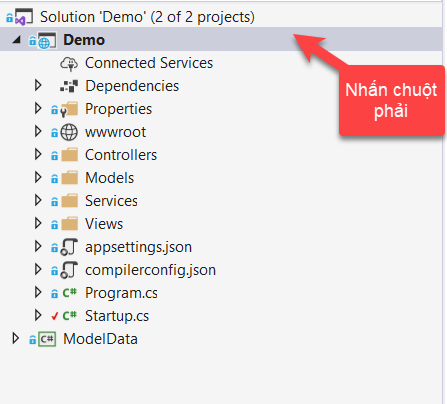
* Ở đây chọn Web application



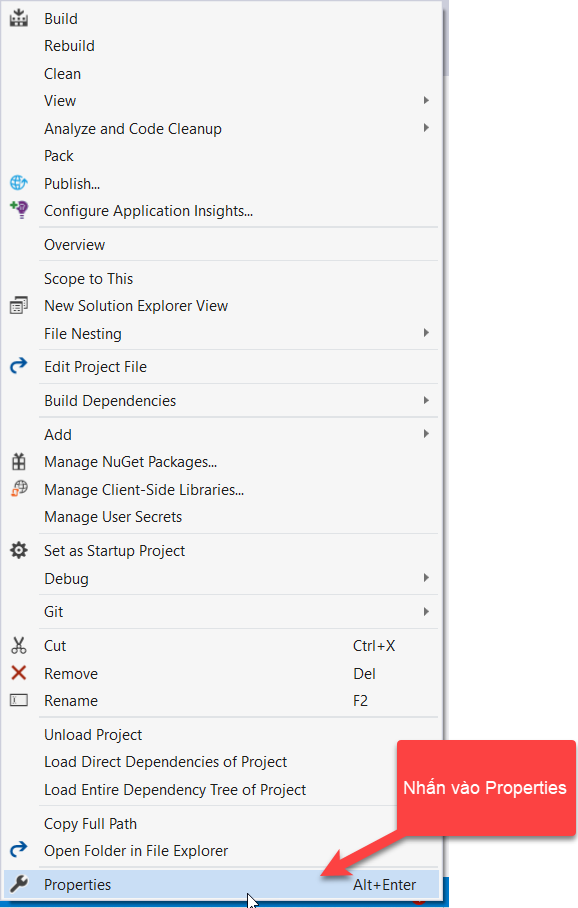
1. Chọn Web application

**Bước 8:** Lấy App URL có trong project để cung cấp cho Google

* Vào Project chọn tên project và nhấn chuột phải

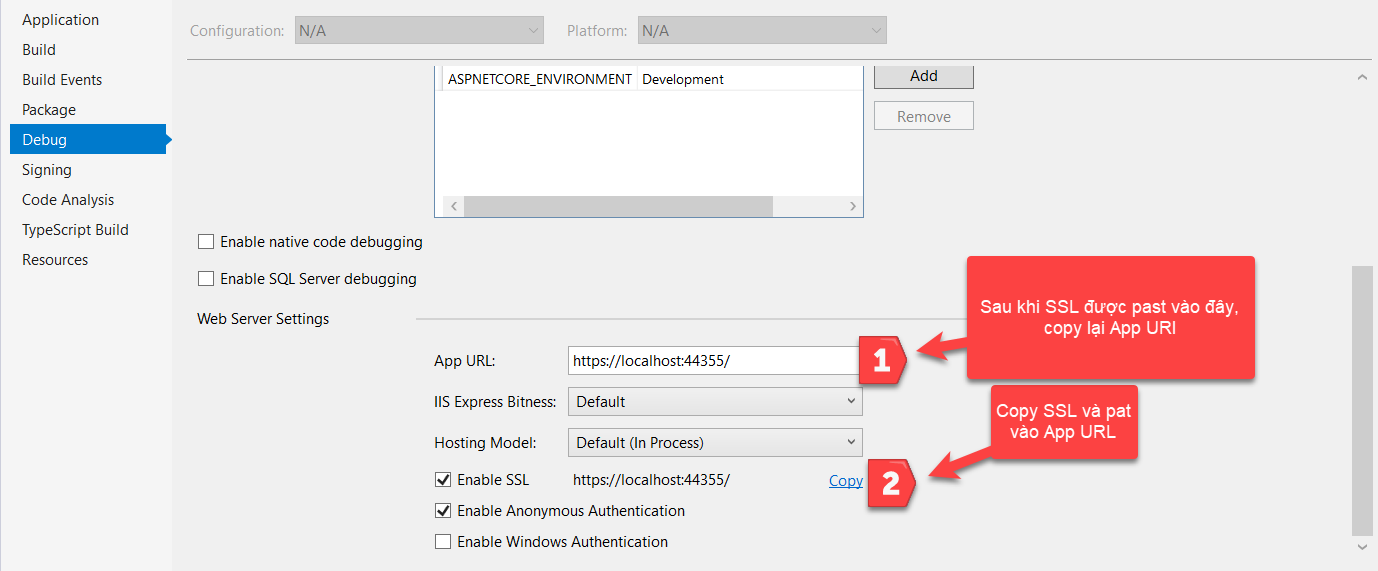


1. Bước lấy App URL trong project

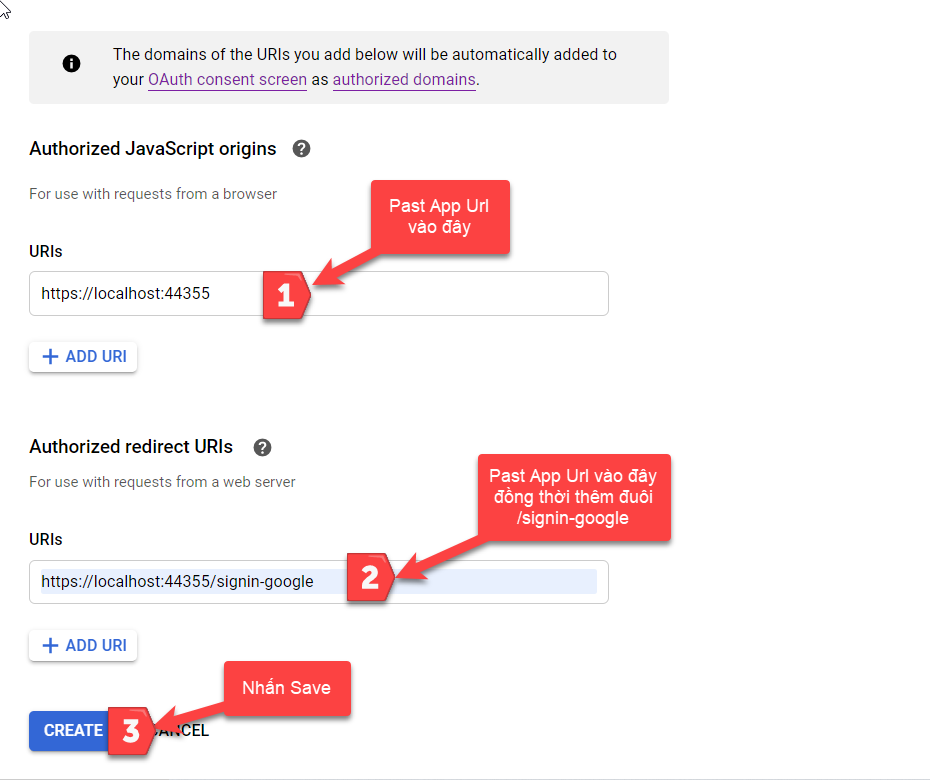


1. Chọn Properties

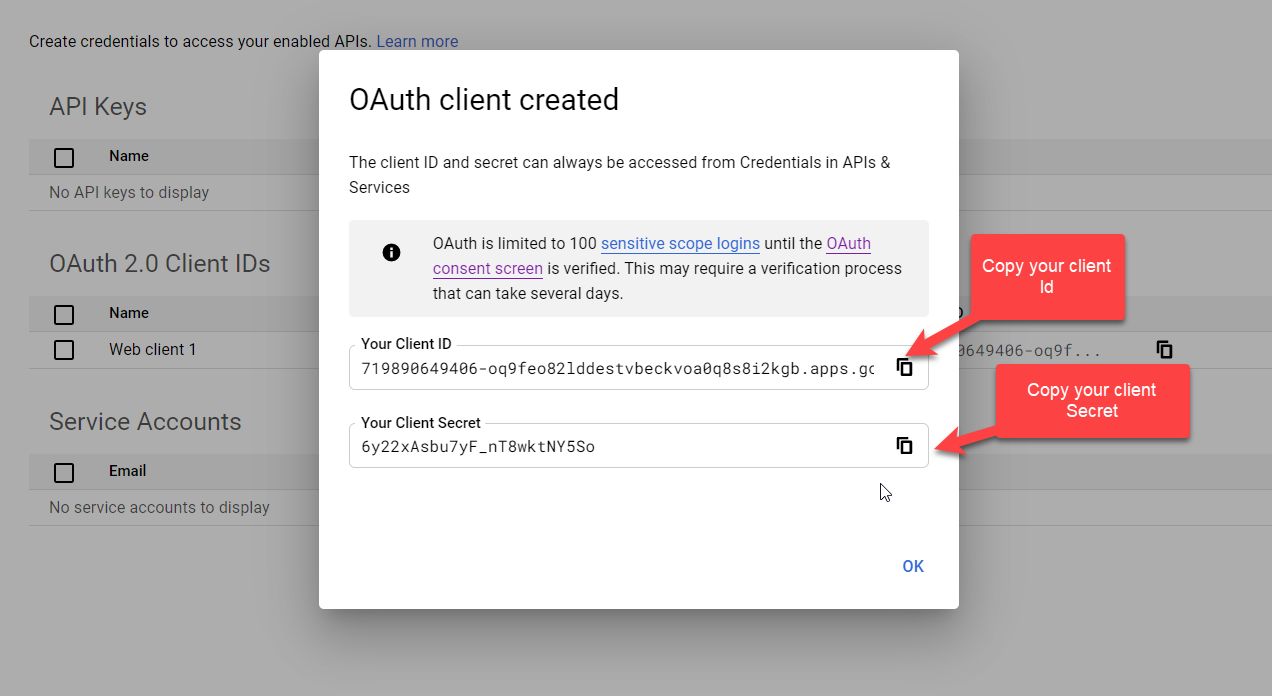
Copy SSL vào trong App URL, sao đó kiểm tra 2 cái đã trùng chưa



1. Copy SSL qua App URL



1. Thiết lập URIs trong Oauth of Google Service

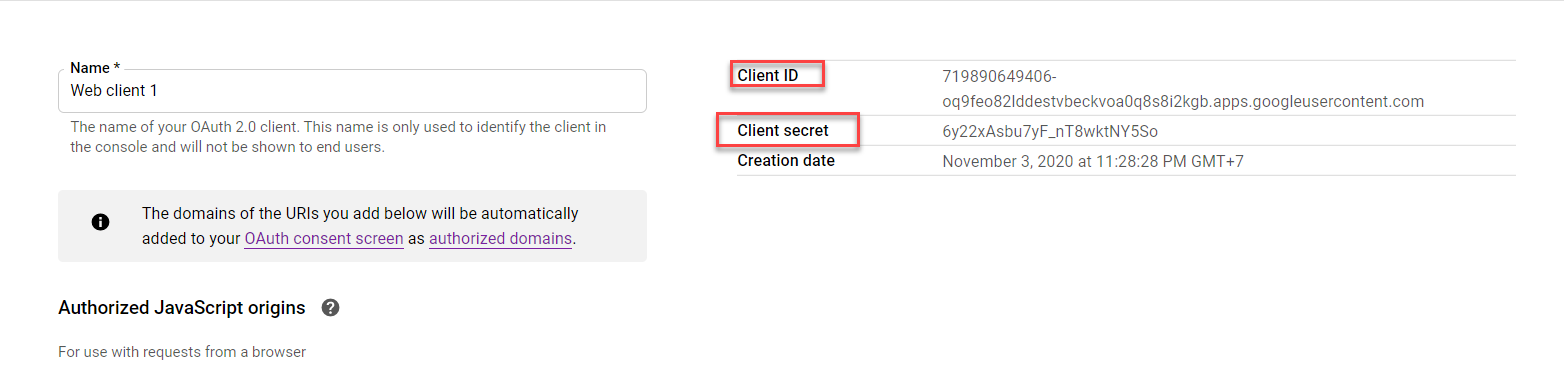


1. Copy Client Id và Client Secret

* Bạn có thể xem Client Id và Client Secret trong ảnh dưới đây



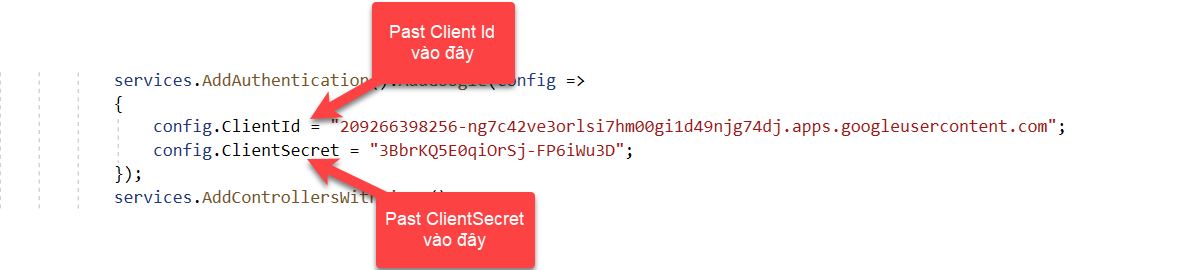
1. Thông tin về Account Oath



1. Thông tin chi tiết về Client Id,Client Secret

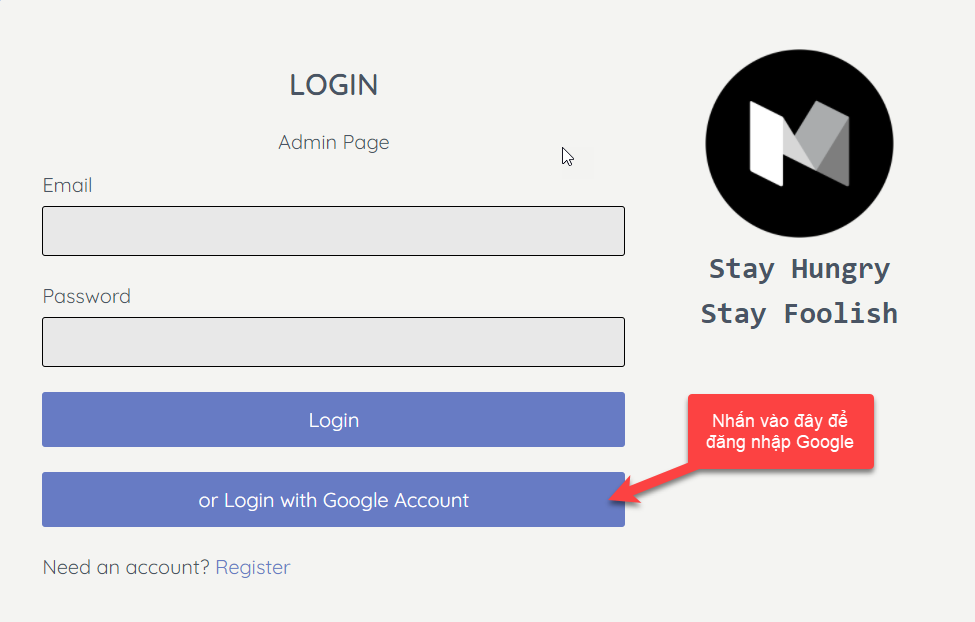
**Bước 9:** Mở lại project vào vào file startup.cs

Copy Client Id và Secret Id đã được cung cấp vào đây



1. Mô tả Client Id và Client Secret

**Bước 10:** Chạy project và đăng nhập tử vào Google



1. Trang login