**Cơ chế của API Repostory.**

**A, Các thành phần của API**

**I, Program.cs**

**1, Khởi tạo ‘builder’ bằng WebApplication.CreataBuilder(args);**

**- WebApplication:** Ứng dụng web được sử dụng để định cấu hình đường dẫn HTTP và các tuyến đường.

- CreateBuilder: Khởi tạo phiên bản mới của lớp Microsoft.AspNetCore cho WebApplication với các giá trị mặc định được định cấu hình trước

=> Khỏi tạo ‘builder’ làm 1 ứng dụng web cấu hình được dẫn HTTP mới được cấu hình với giá trị của ASP.NetCore

**2, Thêm dịch vụ vào vùng chứa**. (builder.Service.

**- Service:** Một tập hợp các dịch vụ cho ứng dụng để soạn thảo. Điều này rất hữu ích để thêm các dịch vụ do người dùng cung cấp hoặc do khuôn khổ cung cấp.

- **AddController:** Thêm các dịch vụ cho bộ điều khiển được chỉ định. Phương thức này sẽ không đăng ký các dịch vụ được sử dụng cho các lượt xem hoặc trang.



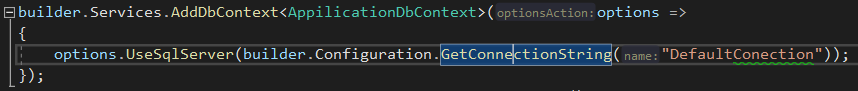
Hình 1

Hình 1: Câu lệnh Thêm các dịch vụ cần thiết cho bộ điều khiển được chỉ định.

- **AddDbContexxt:** Sử dụng phương pháp này khi sử dụng chèn phụ thuộc(DI) trong ứng dụng. Giúp cho quá trình truy vấn lấy dữ liệu bất đồng bộ => Có thể nhận nhiều yêu cầu truy vấn cùng lúc và từ nhiều điểm yêu câu khác nhau.

- **UseSQLServer**: dùng để Định cấu hình ngữ cảnh để kết nối với cơ sở dữ liệu Microsoft SQL Server.

- **Configuration:** Tập hợp các Cấu hình được cung cấp trong ứng dụng.

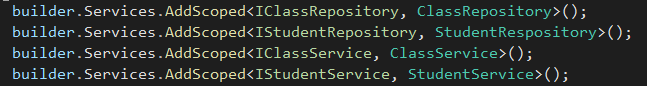


Hình 2

- Hình 2: dùng để Nhận các cấu hình được cài đặt cho Entities trong Class “AppilicationDbContext” để truyền xuống cho Cơ Sở Dữ Liệu tạo mới 1 DB có cấu hình các thuộc tính và mối quan hệ giữa các Bảng và Ánh xạ lên Code. Đồng thời, Hình 2 dùng để xác định nơi lưu trữ dữ liệu và tạo ra 1 (DI) để có thể gọi, truy vấn và thao tác với dữ liệu ở các phương thức trong Ứng dụng.

- AddScoped<T>(): Thêm dịch vụ T vào. Mỗi khi gọi đến thì sẽ được sử dụng các hàm, phương thức trong phạm vi Class T quản lý.

- Tương tự, với AddTransient và AddSingleton nhưng khác nhau về thời gian sử dụng.



Hình 3

- Hình 3: thêm các DI của các Cần thiết vào ứng dụng để sử dụng các các dịch vụ mà các Class trên đang chứa. khi cần thiết thì có thể Inject vào các Class, Controller cần sử dụng.

**II. Entities**

- **Entities :** nơi chứa các thực thể ánh xạ với Cơ Sở Dữ Liệu.

- Các thực thể này có các thuộc tính tương đương với các cột dữ liệu trong Bảng dữ liệu mà thực thể đó Ánh Xạ.

- Các thực thể có thể được cấu hình quan hệ ánh xạ trực tiếp bằng KeyWord chú thích dữ liệu tương ứng với các thuộc tính cần được không chế hoặc liên kết trong CSDL.

Cách Sử dụng: tham khảo thêm tại: <https://www.entityframeworktutorial.net/code-first/dataannotation-in-code-first.aspx>

**II. Context- ApplicationDbContext.cs**

**ApplicationDbContext.cs:** Dùng để:

1, Quản lý kết nối cơ sở dữ liệu.

2, Cấu hình Mô hình và mối quan hệ cho các thực thể.

3, Truy vấn Cơ Sở dữ liệu.

4, Lưu trữ dữ liệu vào Cơ sở Dữ liệu.

5, Cấu hình các thay đổi của dữ liệu.

6, Quản lý giao dịch dữ liệu giữa CSDL và ứng dụng.

Note: Có thể dùng Phương thức OnMoreCreating hoặc tạo file **Configuration** để đặt điều kiện cho các thuộc tính, cấu hình mối quan hệ cho các thực thể bằng Fluent API.

- Cách sử dụng Fluent API có thể tham khảo thêm tại: <https://www.entityframeworktutorial.net/efcore/fluent-api-in-entity-framework-core.aspx>

**III, Respositoties**

**1, File Implement**

**- Implement :** Nơi chứa Lớp có các hoạt động do EF API xây dựng và thực thi các lệnh INSERT, UPDATE và DELETE, dựa trên trạng thái của một thực thể. SaveChange() dùng để lưu hoạt động trên vào cơ Sở dữ liệu và dùng để truy vấn dữ liệu từ cơ sở dữ liệu lên cho ứng dụng.

**2, File Interface**

**- Interface:** nơi chứa các Lớp giao diện của các lớp thực hiện chức năng cho các Entity.

**IV, View Models**

**1, File Models Show**

**- ModelsShow** chứa các object có các thuộc tính cần thiết khi gửi lên hiển thị cho người dùng.

**- ModelsCreate** nơi chứa các object có các thuộc tính cần thiết truyền vào để xử lý dữ liệu khi người dùng muốn thêm, sửa, xóa dữ liệu trong Cơ sở dữ liệu.

**IV, Service**

**1, File Implement trong Service**

**\* File Implement trong Service :**  nơi chứa các class **thực hiện** các chức năng **truy vấn** và **chuyển đổi dữ liệu** sao cho phù hợp để chuyển lên GUI. Đồng thời, **nhận** các **dữ liệu truyền vào** và **yêu cầu** Thêm, Sửa, Xóa dữ liệu trong Cơ sở dữ liệu.

**2, File Interface trong Service**

**\* File Interface trong Service:** Chứa các lớp giao diện của lớp chức năng bên Implement Service.

**V, File Controller**

**\* File Controller** chứa các Class Controller.

\* **Controller**: nơi đầu tiên nhận các Request từ người dùng. Khi người dùng truy cập URL trên trình duyệt. ASP.Net Core sẽ nhận các Resquet thông qua Routing và gọi đến Controller cụ thể.

- Ex: <https://localhost:7218/api/Class/GetList>

=> Resquet gọi đến Controller có tên là ClassController và gọi đến phương thức thực hiện tên là GetList.

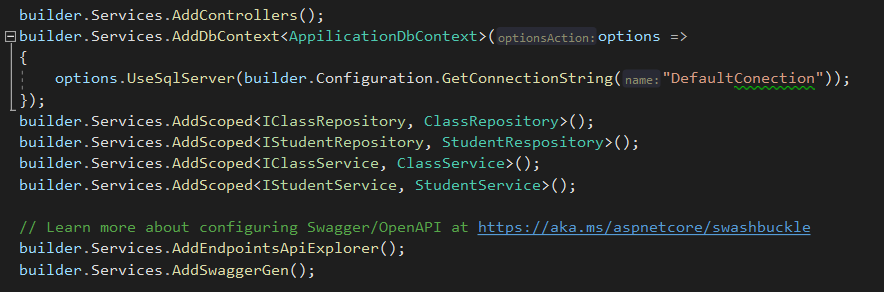
\* Trong Controller, trước mỗi action method đều có Http+ các động từ như Get, Post, Put, Delete, Parth,… để bộ điều khiển Web API được gọi đến và Xác định Action Method được thực thi khi người dùng gửi yêu cầu.

**B, Luồng hoạt động của API Respository**

**I, Khi ứng dụng bắt đầu chạy.**

**1, Program.cs**

**-** Khi chương trình bắt đầu chạy, từ Program khởi tạo mới ứng dụng. Thêm vào ứng dụng các dịch vụ cần thiết.

Ex: 

Hình 4

- Ở Hình 4, ứng dụng được thêm các dịch vụ về Dependentcy để tạo các sự phụ thuộc được xây dựng ban đầu cho các Class chức năng cần thiết để tạo sự kiên kết nhẹ.

- Đồng thời, ứng dụng được thêm các dịch vụ để định cấu hình cho các Controller và cấu hình ApiExplorer.

Tiếp theo, ứng dụng được xây dựng : 

Sau đó, tiếp tục cài đặt Cấu hình đường dẫn HTTP.



1, Thêm phần mềm trung gian để chuyển hướng Yêu cầu HTTP sang HTTPS.

2, Chỉ định và cấp quyền cho một tài nguyên được định tuyến bằng cách sử dụng định tuyến điểm cuối.

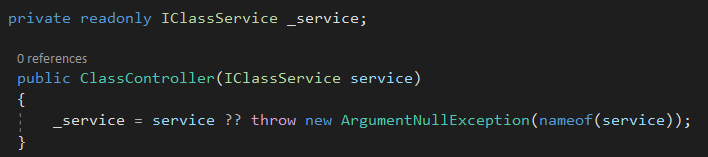
3, Thêm điểm cuối cho các hành động của bộ điều khiển.

4, Chạy ứng dụng và chặn chuỗi Resquet cho đến khi máy chủ tắt.

**II, Luồng chạy khi Ứng dụng nhận được Các Resquet của User**

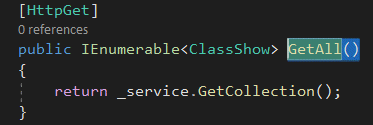
**1, Người dùng gửi Resquet để lấy dữ liệu.**

**-** Trong mỗi Controller được tiêm sự phụ thuộc ( Inject) của lớp Service tương ứng để sử dụng các Phương thức bên trong Phục vụ cho chức năng.

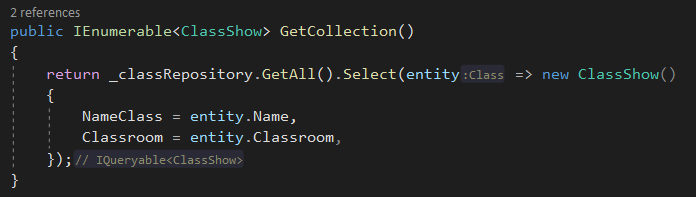


**Ví dụ: ClassController**

**-** Khi người dùng gửi yêu cầu cần lấy thông tin các lớp học từ cơ sở dữ liệu thì trên URL của Trình duyệt <https://localhost:7218/api/Class/GetAll()>. Resquet gọi xuống ClassController để truy vấn dữ liệu qua phương thức GetAll() để lấy dữ liệu được xác định qua bộ định tuyến [HttpGet] để thực hiện Action Method GetAll() trả về kiểu dữ liệu IEnumerable<ClassShow>.



- Bên trong Phương thức GetAll() gọi đến Phương thức GetCollection() của ServiceClass.



- PHương thức GetCollection() gọi đến PT GetAll() của lớp ClassRepostory để truy vấn dữ liệu từ DB và chuyển đổi sang kiểu dữ liệu đối tượng ClassShow. Để hiển thị thông tin về các các Lớp học hiện có trong Có trong Cơ sở dữ liệu. và được truyền lên hiển thi cho người dùng