**Migration ASP.NET**

# Tổng quan :

Trong các dự án trong thế giới thực, các mô hình dữ liệu thay đổi khi các tính năng được triển khai: các thực thể hoặc thuộc tính mới được thêm và xóa và các lược đồ cơ sở dữ liệu cần được thay đổi tương ứng để được đồng bộ hóa với ứng dụng. Tính năng di chuyển trong EF Core cung cấp một cách để cập nhật dần lược đồ cơ sở dữ liệu nhằm giữ cho lược đồ này đồng bộ với mô hình dữ liệu của ứng dụng trong khi vẫn bảo toàn dữ liệu hiện có trong cơ sở dữ liệu.

Ở cấp độ cao, quá trình di chuyển hoạt động theo cách sau: Khi thay đổi mô hình dữ liệu được đưa ra, nhà phát triển sử dụng các công cụ của EF Core để thêm một quá trình di chuyển tương ứng mô tả các bản cập nhật cần thiết để giữ cho lược đồ cơ sở dữ liệu được đồng bộ hóa.EF so sánh với sơ lược rút gọn của mô hình cũ để xác định được sự khác biệt và tạo các tệp Migration , Các tệp Migration đều có thể được theo dõi trong phần source control của dự án giống như các phần tệp khác. Khi một lần di chuyển mới đã được tạo, nó có thể được áp dụng cho cơ sở dữ liệu theo nhiều cách khác nhau. EF Core ghi lại tất cả các lần di chuyển đã áp dụng trong một bảng lịch sử đặc biệt, cho phép nó biết lần di chuyển nào đã được áp dụng và lần di chuyển nào chưa.

Trong quá trình phát triển , bạn có thể đã sử dụng API tạo và hủy lặp đi lặp lại nhanh chóng . Nhưng bây giờ nếu dự án được sản xuất thì bạn cần nâng cấp lược đồ một cách an toàn để tránh bị thất thoát dữ liệu

**Cài đặt môi trường và công cụ :**

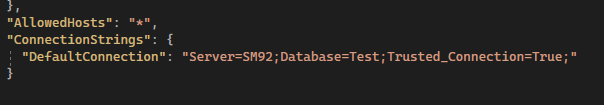
Microsoft.EntityFrameworkCore : FrameWork cho phép làm việc với cơ sở dữ liệu bằng các đối tượng .NET ( [Overview of Entity Framework Core - EF Core | Microsoft Learn](https://learn.microsoft.com/en-us/ef/core/) )

Microsoft.EntityFrameworkCore.Tools : Công cụ cần thiết để cho phép tạo Migration bằng các dòng lệnh

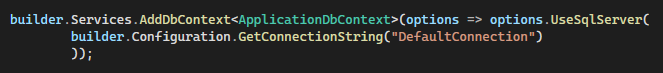
Microsoft.EntityFrameworkCore.SqlServer ( Optional ) : Kết nối cơ sở dữ liệu , Package ở đây là dùng với SqlServer , thay đổi theo nhu cầu ([Database Providers - EF Core | Microsoft Learn](https://learn.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli))

Các bước kết nối database :

Trong file appsetting.json : thêm ConnectionString cho App

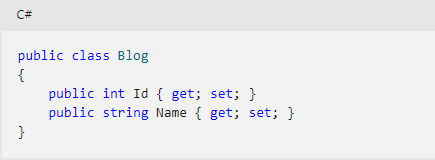


Trong program.cs : Hãy thêm GetConnectionString cho App (trong trường hợp này “options” sẽ là SQL Server database) đây là bước tất yếu để cho phần mềm hiểu là nó sẽ lấy cấu trúc data ở đâu , nếu không các hoạt động sau sẽ không được thực thi



# Cách dùng Migration:

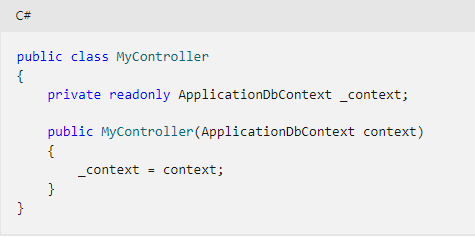
Giả sử bạn vừa hoàn thành ứng dụng EF Core đầu tiên của mình, ứng dụng này chứa mô hình đơn giản sau:



Trong trường hợp tạo migration bạn nên tạo thêm một lớp con của DbContext gọi là ApplicationDbContext như là một dịch vụ có phạm vi chủ yếu cung cấp cho ASP.NET Core . Nó được cấu hình để nhà cung cấp SQL Server sử dụng và đọc chuỗi kết nối từ cấu hình ASP.NET Core . Lớp ApplicationDbContext phải hiển thị hàm tạo công khai với tham số DbContextOptions<ApplicationDbContext>. Đây là cách cấu hình ngữ cảnh từ AddDbContext được chuyển đến DbContext.



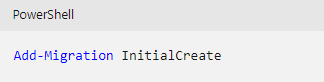
ApplicationDbContext sau đó có thể được sử dụng trong Controller ASP.NET Core hoặc các dịch vụ khác thông qua hàm tạo



Kết quả cuối cùng là một phiên bản ApplicationDbContext được tạo cho mỗi yêu cầu và được chuyển đến Controller để thực hiện từng công việc trước khi được xử lý khi yêu cầu kết thúc.

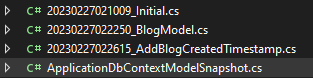
Sau khi đã tạo được Model thì bước tiếp theo thì ta sẽ tạo một class Migration tương ứng cho Model đó.

Các bạn vào Tool à Nuget Package Manager à Package Manager Console .



Bấm Add-Migration + “ Tên tự đặt ”

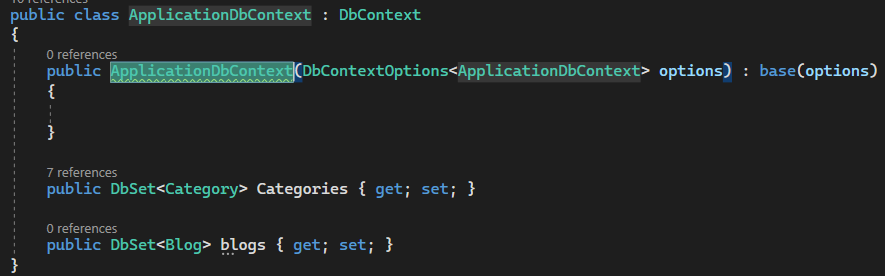
Đợi vài giây và Project sẽ tự động tạo một thư mục Migration có class Migration vừa tạo và một bảng tóm tắt ngắn gọn lịch sử đã tạo các Migration

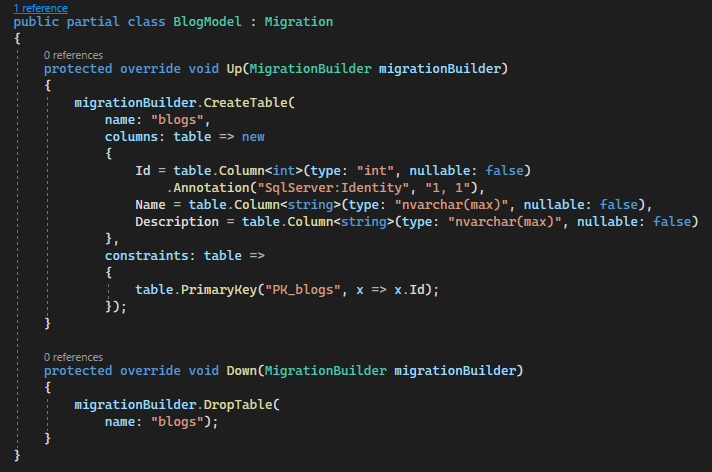


Sau đó bạn hãy kiểm tra Migration vừa tạo xem đã được ưng ý chưa , nếu không bạn có thể bấm Remove-Migration .

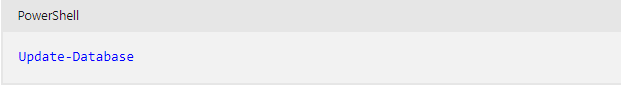


Nếu Migration trả về một Migration rỗng và trong hàm Up và Down không có một logic gì , thì hiện bạn đang gặp lỗi trong việc detect async của Database hãy check lại kỹ xem mình đã thêm Model đó vào ApplicationDbContext.cs chưa ?

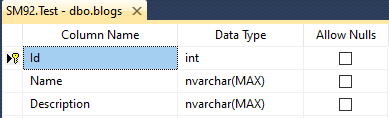




Nếu Migration được trả về như thế này thì bạn đã thành công trong bước tạo Migration .  
Tiếp theo hãy tạo update lên Database đã được gọi trước đó .

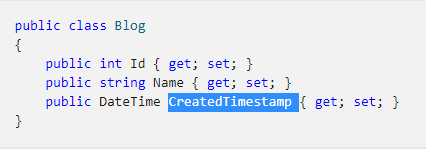


Đợi vài phút thì khi nếu nó báo Done thì ta lên kiểm tra SQL Server thì thấy Model Blog đã được nhận .

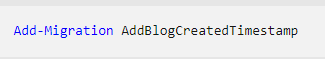


# Cải tiến Model :

Giả sử sau một vài ngày và bạn phải thêm vào một trong cho Model của mình và trong trường hợp ở đây là CreatedTimestamp

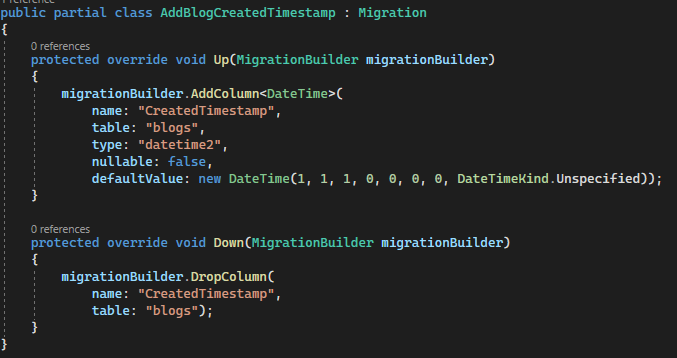


Mô hình của bạn và cơ sở dữ liệu của bạn hiện không đồng bộ - chúng ta phải thêm một cột mới vào lược đồ cơ sở dữ liệu của bạn. Hãy tạo một Migration mới cho nó :

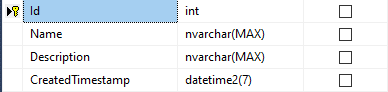


Lưu ý rằng chúng tôi cung cấp cho quá trình di chuyển một tên mô tả để dễ hiểu lịch sử dự án sau này.

Vì đây không phải là lần di chuyển đầu tiên của dự án, EF Core hiện sẽ so sánh mô hình được cập nhật của bạn với phần sơ lược của mô hình cũ, trước khi cột được thêm vào; phần sơ lược mô hình là một trong những tệp được tạo bởi EF Core khi bạn thêm một lần di chuyển và có thể kiểm tra được trong source code. Dựa trên sự so sánh đó, EF Core phát hiện ra rằng một cột đã được thêm vào và thêm phần Migration thích hợp.



Sau khi đã thêm Migration thành công thì ta bấm “ Update-database ” để thêm Migration mới đó vào database . Đợi vài giây và khi nó báo success thì hãy kiểm tra trên SQL Server .



Phần CreatedTimestamp đã thêm thành công .

**Tham khảo thêm về cách seeding data bằng Entity FrameworkCore**

[Migrations and Seed Data With Entity Framework Core - Code Maze (code-maze.com)](https://code-maze.com/migrations-and-seed-data-efcore/)

**Tài liệu tham khảo :**

[Migrations Overview - EF Core | Microsoft Learn](https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli)