# Tuần 1

## Cấu trúc dữ liệu cây cân bằng

## Tìm Kiếm theo index clustered-và tối ưu cách đánh (Cây chỉ muc)

- Đọc dữ liệu nhanh hơn

- Tốc độ câu Query nhanh hơn

- Update,insert,delete dữ liệu lâu hơn

- Ưu tiên tạo index cho những cột có tần suất select nhiều

- Được Sử dụng tự động bởi database engine

- Có 2 Loại index

+ Clusteted:

* Thường tạo chung với khoá
* Tối đa 1 Clustered index trên 1 table
* Sắp xếp trực tiếp theo 1 cột

+ NonClustered

* Sắp xếp dữ liệu theo tiêu chí mh đưa ra
* Có thể có nhiều non-Clustered index trên 1 table
* Thường được tạo bằng lệnh hay hoặc đi cùng với unique-key

## 3. Tìm hiểu về function

-Có 2 loại:

+ Trả về giá trị đơn

-------------------------------------

-------------------------------------

GO

CREATE FUNCTION fn\_TongTienTheoHD

(

@MaHD INT

)

RETURNS FLOAT

AS BEGIN

DECLARE @Tong FLOAT

SELECT @Tong = SUM(SoLuong\*DonGia\*(1-GiamGia))

FROM ChiTietHD

WHERE MaHD = @MaHD

RETURN @Tong

END

GO

-------------------------------------------------

--------------------------------------------------------

SELECT dbo.fn\_TongTienTheoHD(10250)

+Trả về bảng

## 4. Store Procedure giống khác nhau với function

* Cả stored procedure và function đều là các đối tượng cơ sở dữ liệu chứa một tập các câu lệnh SQL để hoàn thành một tác vụ.
* Một stored procedure (thủ tục lưu trữ) có thể sử dụng lại nhiều lần. Vì vậy, nếu bạn có một truy vấn SQL mà bạn có ý định sử dụng nhiều lần thì hãy lưu nó dưới dạng một thủ tục lưu trữ, sau đó chỉ cần gọi nó để nó thực thi truy vấn SQL của bạn. Ngoài ra, bạn củng có thể truyền tham số cho một thủ tục lưu trữ...
* Một function (hàm) được biên dịch và thực thi mỗi khi hàm đó được gọi. Hàm phải trả về giá trị...

- Sự khác nhau cơ bản giữa store procedure và function

* Thủ tục lưu trữ có thể trả về giá trị zero, một hoặc nhiều giá trị. Trong khi hàm phải trả về một giá trị duy nhất (có thể là bảng).
* Các hàm chỉ có thể có các tham số đầu vào cho nó trong khi thủ tục lưu trữ có thể có các tham số đầu vào hoặc đầu ra.
* Hàm có thể được gọi từ thủ tục lưu trữ trong khi thủ tục lưu trữ không thể được gọi từ hàm.

- Sự khác biệt nâng cao giữa Stored Procedure và Function

* Các thủ tục không thể được sử dụng trong câu lệnh SELECT trong khi hàm có thể được nhúng trong câu lệnh SELECT. Bởi vì một thủ tục có thể trả về nhiều tập kết quả nên nó không phù hợp để sử dụng trong câu lệnh SELECT.
* Các thủ tục lưu trữ không thể được sử dụng trong các câu lệnh WHERE / HAVING / SELECT trong khi hàm thì có thể.
* Một ngoại lệ có thể được xử lý bằng TRY-CATCH trong thủ tục lưu trữ, đối với hàm thì không thể.
* Có thể sử dụng TRANSACTIONS trong thủ tục lưu trữ, với hàm thì không thể.

## 5. SubQuery

+ Sử dụng khi không thể sử dụng toán tử join đk nữa, hoặc toán tử join quá phức tạp

+ Phép toán join chạy nhanh hơn 500 lần so với subquery

## 6. Trigger

+ Không có tham số truyền vào

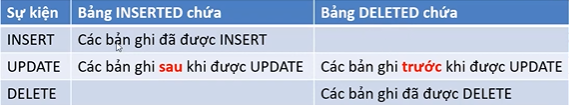
+gắn vưới mỗi bảng cụ thể

+Thực hiện tự động khi xảy ra các sự kiện Insert, update, Delete

+Trong trigger tồn tại 2 bảng đặc biệt

----Inserted

----Deleted



# Tuần 2 net core:

## So sánh asp.net core và net framwork

* ASP.NET core là 1 tập hơn thư viện quy chuẩn giống như 1 famwork để thiết kế 1 web size
* Gọn nhẹ hơn, Linh hoạt hơn
* Chạy được trên cả notnet core và notnet farmwork
* Notnet 5 gồm 2 nền tảng notnet core và notnet farmwork
* Notnet core hỗ trợ đa nền tảng,farmwork chạy trên windown

## net core có thành phần gì : các thư mục startup, controller, app config

- Properties chứa các profile để thực thi ứng dụng, chứa thông tin cấu hình của ứng dụng

- Appsetting.json: chứa thông tin cấu hình cho hoạt động của ứng dụng như connectionstring, các biến mỗi trường, các tham số dòng lệnh

- launchSettings chỉ dùng cho local

- Dependencies chưa các gói nuget mà chúng ta sử dụng

- Program tạo 1web server với main method của nó

- Confic các services cần thiết cho ứng dụng để sử lý các request

- PROGRAM tạo hostting và là đầu vào đầu tiên của ứng dụng

- Tất cả các ứng dụng .NET Core đều là ứng dụng Console. Các loại ứng dụng khác như MVC, SPA, v.v. được xây dựng trên ứng dụng bảng điều khiển.

• Ứng dụng Console bắt đầu bằng một program.cs, Program này phải chứa một phương thức static void Main.

• Phương thức này được gọi bất cứ khi nào ứng dụng khởi động.

• Đây là điểm vào ứng dụng đầu tiên

File này chứa class program chịu trách nhiệm cấu hình nền tảng() của ứng dụng

Class này cũng chưa entity point của ứng dụng

- Startup

• Lớp startup là một lớp đơn giản, không kế thừa từ bất kỳ thứ gì cũng như không triển khai bất kỳ implemening nào

• Nó có hai chức năng chính.

• ConfigureServices Phương thức này được gọi bởi thời gian chạy. Sử dụng phương pháp này để thêm dịch vụ vào vùng chứa.

• Configures Phương thức này được gọi bởi thời gian chạy. Sử dụng phương pháp này để định cấu hình request pipeline HTTP.

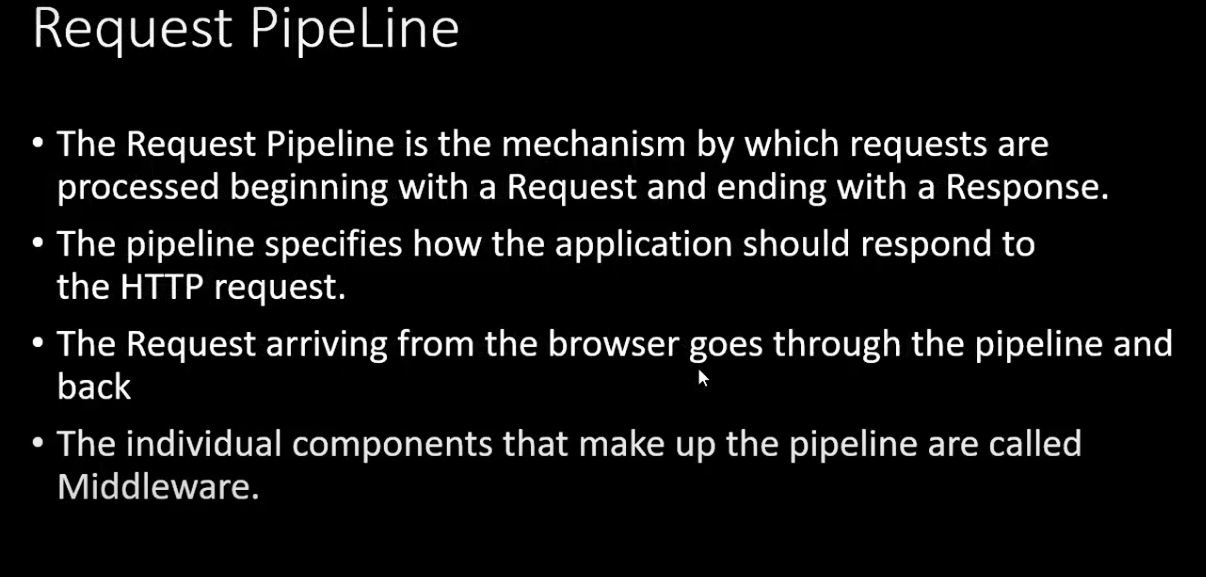
* Startup.cs: File này chứa class với các phương thức cấu hình cho hoạt động của ứng dụng, ví dụ bạn sẽ sử dụng những middleware nào, thứ tự sắp xếp các middleware trong pipeline ra sao. Bạn cũng có thể cấu hình sử dụng các loại dịch vụ (service) nào, như Dependency Injection, Logging.
* Wwwroot: chưa các file tĩnh như html,css, javascrip,hình ảnh. Các file tĩnh phải nằm trong thư mục này thì mới có thể cung cấp cho trình duyệt

## Configuration và configuration service là gì

* Configuration: Cấu hình các request pipeline và middleware
* ConfigureServices là nơi bạn có thể đăng ký các lớp phụ thuộc của mình với vùng chứa IoC tích hợp sẵn. Sau khi đăng ký lớp phụ thuộc, nó có thể được sử dụng ở bất kỳ đâu trong ứng dụng. Bạn chỉ cần đưa nó vào tham số của hàm tạo của một lớp mà bạn muốn sử dụng nó. Vùng chứa IoC sẽ tự động đưa nó vào.

## Request pipeline

* Cách ứng dụng lm vc với các request gồm các bước nào



Đường ống yêu cầu

* Request Pipeline là cơ chế mà các yêu cầu được xử lý bắt đầu bằng Yêu cầu và kết thúc bằng Phản hồi.
* Pipeline chỉ định cách ứng dụng sẽ phản hồi yêu cầu HTTP
* Yêu cầu đến từ trình duyệt đi qua pipeline và quay lại
* Các thành phần riêng lẻ tạo nên Pipeline được gọi là Middeware



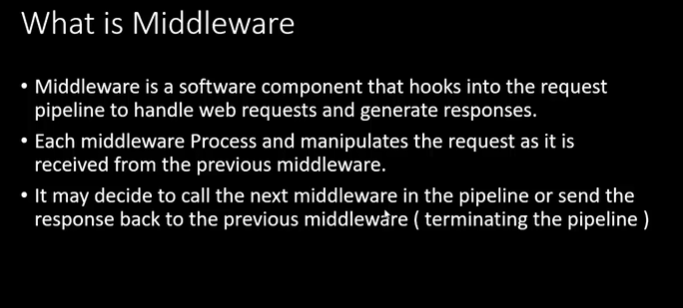
Định cấu Request Pipeline

* Định cấu hình Middleware bằng cách sử dụng và chạy
* Vấn đề đặt hàng
* Custom Middleware
* Tạo Middleware tùy chỉnh
* Phương pháp mở rộng

## Middleware

Là nhóm thành phần có nhiệm vụ sử lí truy vấn HTTP, các middleware được xếp theo chuỗi. Khi truy vấn chạy qua mỗi khâu của chuỗi middle sẽ được xem xét sử lí

-HTTP SERVER , còn gọi là máy chủ được xây dựng với tên gọi Kestrel, có nhiệm vụ tiếp nhận vấn đề HTTP. Kestrel có thể hoạt động độc lập (tích hợp trong một ứng dụng khác) hoặc phối hợp với một máy chủ web thông thường (Apache, NGinx, IIS).

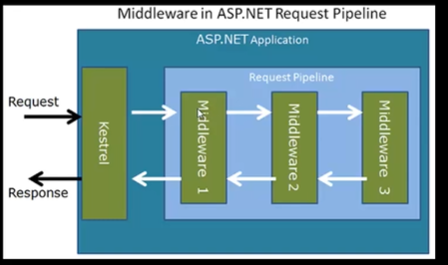


Middleware là gì ?

• Middleware là một thành phần phần mềm kết nối vào request pipeline để xử lý các yêu cầu web và tạo phản hồi.

• Mỗi Middleware Xử lý và thao tác yêu cầu khi nó được nhận từ Middleware trước đó.

• Nó có thể quyết định gọi Middleware tiếp theo trong pipeline hoặc gửi phản hồi trở lại Middleware trước đó (chấm dứt pipeline)



## routing + http request

* Roiting là 1 middleware
* Định tuyến controller sử lí request
* Map các request đến vào controller action,chọn acion tương ứng để sử lí request đó
* Tạo routing output chuyển hướng đến url tương ứng
* Các khái niệm

+ Route collection

+ Route Handler

## action result

* Khái niệm: Action trong ASP.NET Core là những phương thức public của controller có thể được sử dụng để xử lý truy vấn. Action result là cách gọi chung của kết quả thực hiện của action. Kết quả trả về của mỗi action có thể được biểu diễn ở dạng giao diện HTML, ở dạng dữ liệu (XML, JSON) hoặc các dạng thức đặc biệt khác
* Có rất nhiều con: json result,file result
* IActionResult và ActionResult

IActionResult là một interface nó định nghĩa một khuôn mẫu cho toàn bộ các Action Result của một action method. ActionResult là một abstract base class triển khai interface IActionResult. Action result như ViewResult, PartialViewResult hay JsonResult...đều kế thừa từ ActionResult base class.

## model binding

* Model Binding là cơ chế map dữ liệu được gửi qua HTTP Request vào các tham số của action method trong Controller. HTTP Request có thể chứa dữ liệu từ nhiều định dạng. Dữ liệu có thể chứa trong HTML Form. Nó có thể là một phần của route value hoặc trên query string hay có thể là một body của request.
* Cơ chế ASP.NET Core model binding cho phép chúng ta dễ dàng bind các giá trị này vào các tham số của action method. Các tham số này có thể là kiểu nguyên thủy hoặc kiểu đối tượng phức tạp.

## model validtion

* Form Data post dữ liệu lên Controller action tự động được map vào các tham số của action bởi Model Binder.
* Model cần kiểm tra dữ liệu đầu vào xem có 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.
* Cơ chế validation phía client (client-side validation) rất quan trọng vì nó giúp tăng trải nghiệm người dùng khi kiểm tra dữ liệu mà không cần chờ đến server nhưng phía server lại nên đảm bảo một lần nữa để các dữ liệu không hợp lệ không thể đưa vào hệ thống.

## Responsitory

* Responsitory pattern là một phần abstraction của Lớp truy cập dữ liệu.
* Nó ẩn chi tiết về cách chính xác dữ liệu được lưu hoặc truy xuất từ ​​nguồn dữ liệu cơ bản.
* Chi tiết về cách dữ liệu được lưu trữ và truy xuất có trong Repository Pattern tương ứng.
* Ví dụ: bạn có thể có một kho lưu trữ và truy xuất dữ liệu từ một bộ sưu tập trong bộ nhớ. Bạn có thể có một Responsitory lưu trữ khác lưu trữ và truy xuất dữ liệu từ cơ sở dữ liệu như SQL Server. Tuy nhiên, một Repository Pattern khác lưu trữ và truy xuất dữ liệu từ tệp XML.

**Repository Pattern Interface**

**Mẫu interface trong repository pattern**

* Những operations nào (tức là các methods) được hỗ trợ bởi Repository
* Dữ liệu cần thiết cho mỗi operations tức là các tham số cần được truyền cho methods và dữ liệu mà method returns
* Repository interface chứa những gì nó có thể làm, nhưng không, nó làm như thế nào, nó có thể làm gì
* Chi tiết triển khai nằm trong lớp repository tương ứng implements repository Interface