**- SQL:**

**• Index:**

Index là là đánh chỉ mục trong sql nhằm tăng tốc độ truy vấn và tìm kiếm dữ liệu

* Cluster index: chi co 1 custer index tren 1 table or view
* Non-cluster index: co the co nhieu

CREATE INDEX index\_name  
 ON table\_name (column1, column2, ...);

**• Join:**

Join là phép kết nối 2 hay nhiều bảng lại với nhau. Khi mình cần truy vấn các cột dữ liệu từ nhiều bảng khác nhau để trả về trong cùng 1 tập kqua

Inner join hoặc join lấy về phần chung giữa 2 bảng

Left outer join (left join) là lấy về bảng trái + phần dữ liệu bên phải thỏa mãn điều kiện On

Right outer join (right join) ngược lại

Full outer join (outer join) là lấy tất cả 2 bảng

Cross join

Self join:  sử dụng để thực hiện phép Join trên cùng một bảng bằng cách nối một bảng với chính nó

**• Phân biệt DROP, DELETE, TRUNCATE:**

DELETE: Xóa một hay tất cả dòng trong một bảng theo một điều kiện nhất định, dữ liệu có thể phục hồi lại

TRUNCATE: Xóa toàn bộ các dòng của bảng, giải phóng bộ nhớ và không thể phục hồi lại

DROP: Xóa một bảng khỏi database

**• Phân biệt Where với Having:**

Where: Là câu lệnh điều kiện trả kết quả đối chiếu với từng dòng, được sử dụng để kiểm tra các điều kiện trước khi tập hợp diễn ra.

Having: Là câu lệnh điều kiện trả kết quả đối chiếu cho nhóm (Sum, AVG, COUNT, …), được sử dụng để kiểm tra các điều kiện sau khi quá trình tổng hợp diễn ra

- Vì vậy mà sau GROUP BY thì sẽ chỉ dùng được Having, còn Where thì KHÔNG dùng được sau GROUP BY, HAVING có thể thay thế vị trí dùng cho WHERE. nhưng ngược lại WHERE thì KHÔNG thể thay thế vị trí cho HAVING

**• Subquery:**

Subquery sql là một truy vấn bên trong một truy vấn khác. Nó được sử dụng trong mệnh đề WHERE hoặc HAVING của một câu lệnh SQL. Truy vấn con cho phép chỉ định kết quả của một truy vấn làm đối số trong truy vấn khác

• **Store Procedure:**

Sql Store Procedure là một mã SQL đã chuẩn bị sẵn mà bạn có thể lưu, do đó, mã có thể được sử dụng lại nhiều lần Một truy vấn SQL mà bạn viết đi viết lại nhiều lần, hãy lưu nó dưới dạng một thủ tục được lưu trữ, sau đó chỉ cần gọi nó để thực thi nó

VD: CREATE PROCEDURE SelectAllCustomers  
 AS  
 SELECT \* FROM Customers  
 GO;

* **constraint trong sql**

Là các quy tắc được áp dụng trên cột dữ liệu để kiểm tra tính hợp lệ của dữ liệu đầu vào

Các loại constraint phổ biến: not null, unique, primary key, foreign key, check.

* **Identity**

Sử dụng để để tạo ra cột nhận dạng, chúng chứa các giá trị tự động phát sinh tuần tự theo mỗi hàng trong 1 bảng

| **Tiêu chí** | **Primary key** | **Unique key** |
| --- | --- | --- |
| Công dụng | Nó được sử dụng để làm định danh duy nhất cho mỗi hàng trong bảng. | Nó cũng xác định duy nhất một hàng, nhưng không là khóa chính. |
| NULL | không chấp nhận giá trị NULL | chấp nhận giá trị NULL |
| Số lượng khóa có thể được xác định trong bảng | Chỉ có duy nhất 1 primary key trong 1 bảng | Có thể nhiều hơn 1 |
| Index | Tạo chỉ mục theo nhóm | Tạo chỉ mục không phân cụm |

**- C#**

**• Data Type:**

Là một tập hợp gồm các nhóm loại dữ liệu có cùng đặc tính, cách lưu trữ của dữ liệu và cách thao tác xử lý trên trường dữ liệu đó; nhằm mục đích phân loại các loại dữ liệu.

- Kiểu giữ liệu giá trị: biến đó sẽ được lưu giá trị của đối tượng lưu trữ (float, int, long, …)

- Kiểu giữ liệu tham chiếu: biến chứa địa chỉ trỏ đến nơi lưu đối tượng thực (object, dynamic, string, …)

* **String và StringBuilder**
* String là chuỗi bất biến, khi tạo một đối tượng chuỗi string thì không thể sửa đổi biến đó mà chỉ nó sẽ copy ra một biến mới để thao tác thay đổi
* String Builder là chuỗi có thể thay đổi, khi tạo một đối tượng thì cho phép ta sửa đổi ngay trên biến đó mà ko tạo ra một biến mới như string thường

**• Boxing, unboxing:**

Boxing là quá trình chuyển dữ liệu từ kiểu tham trị sang kiểu tham chiếu, quá trình boxing được thực hiện nhờ quá trình chuyển đổi ngầm định. Unboxing là quá trình ngược lại với Boxing, tức là đưa từ kiểu tham chiếu ra kiểu tham trị. Quá trình này sẽ được thực hiện một cách tường minh.

Khi dùng collection để lưu trữ dữ liệu, những dữ liệu sẽ được lưu trữ kiểu object, nhưng trong một số trường hợp kiểu dữ liệu sẽ là int, long, … Vì vậy quá trình boxing sinh ra là để đồng bộ dữ liệu lưu trữ dưới dạng tham chiếu.

Qúa trình boxing sẽ coppy giá trị của biến giá trị tham chiếu bên bộ nhớ stack lưu sang bộ nhớ heap, sau đó gán địa chị của biến được coppy đó cho một biến bên bộ nhớ stack, nhờ đó biến được coppy giờ là một biến giá trị kiểu tham chiếu

Còn unboxing thì ngược lại

**• Type Casting:**

- Implicit casting(truyền ngầm): được thực hiện tự động khi chuyển loại có kích thước nhỏ hơn sang loại kích thước lớn hơn(char-> int-> long-> float->double)

- Explicit casting(truyền rõ ràng): được thực hiện thủ công, chuyển đổi loại kích thước lớn hơn sang loại kích thước nhỏ hơn(double-> float-> long-> int->char)

**• OOP:**

Lập trình hướng đối tượng là phương pháp lập trình dựa trên khái niệm lớp và đối tượng, tập trung vào các đối tượng thao tác hơn là logic để thao tác chúng

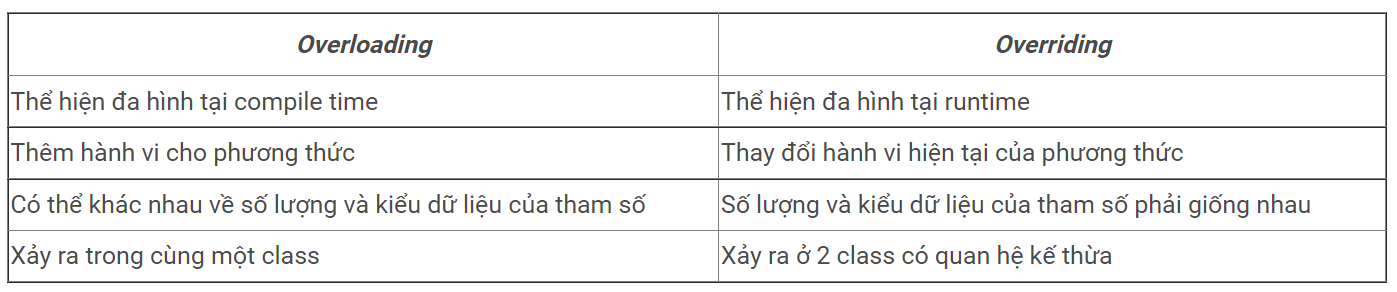
* + - **Tính đóng gói**: Tính đóng gói cho phép che dấu thông tin và những tính chất xử lý bên trong của đối tượng, các đối tượng khác không thể tác động trực tiếp đên dữ liệu bên trong mà bắt buộc phải thông qua phương thức công khai do đối tượng cấp
    - **Tính kế thừa**: cho phép kế thừa lại những thuộc tính, phương thức mà một lớp khác đã có, lớp con kế thừa lại từ lớp cha mà không cần định nghĩa lại, có thể mở rộng các thành phần kế thừa hoặc bổ sung những thành phần mới.
    - **Tính đa hình**: cho phép các đối tượng khác nhau thực thi các chức năng giống nhau theo những cách khác nhau
    - **Tính trừu tượng**: Tính trừu tượng giúp loại bỏ những thứ phức tạp, không cần thiết của đối tượng và chỉ tập trung vào những gì cốt lõi, quan trọng

Chỉ hiển thị ra những thông tin cần thiết, ấn đi những thành phần không quan trọng

**• override và overload:**

Override ghi đè, thay đổi hành vi hiện tại của một phương thức, còn overload chỉ là mở rộng(thêm) hành vi cho phương thức bằng cách tăng hoặc giảm, thay đổi kiểu giá trị tham số truyền vào

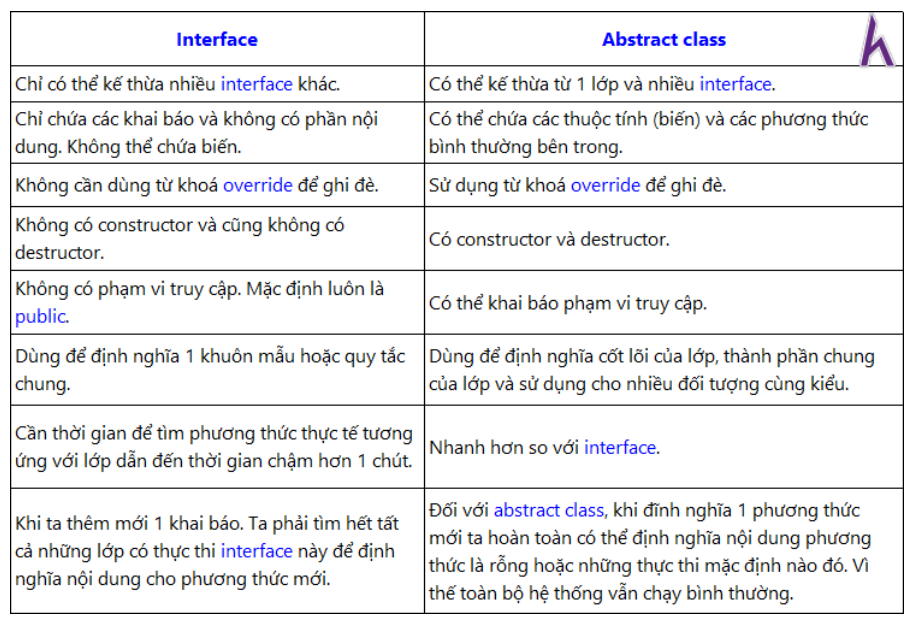
Override: xảy ra ở 2 class có quan hệ kế thừa Overload: xảy ra ở cùng một class



• **Abstract class và Interface**

Abstract là một class cha để cho tất cả các class có cùng thuộc tính, phương thức kế thừa và sử dụng những thuộc tính và phương thức đó còn Interface giống như một chức năng, hợp đồng bạn có thể thêm vào bất kỳ class nào, Interface có thể bao gồm nhiều phương thức.

* Khi nào dùng abstract class, khi nào dùng interface:
  + - Khi chúng ta có nhóm các đối tượng có cùng bản chất (tức là có cùng hành vi và thuộc tính) thì dùng abstract class
    - Khi chúng ta có các nhóm đối tượng khác bản chất nhưng cùng hành vi thì ta dùng interface



• **DataSet và DataTable**

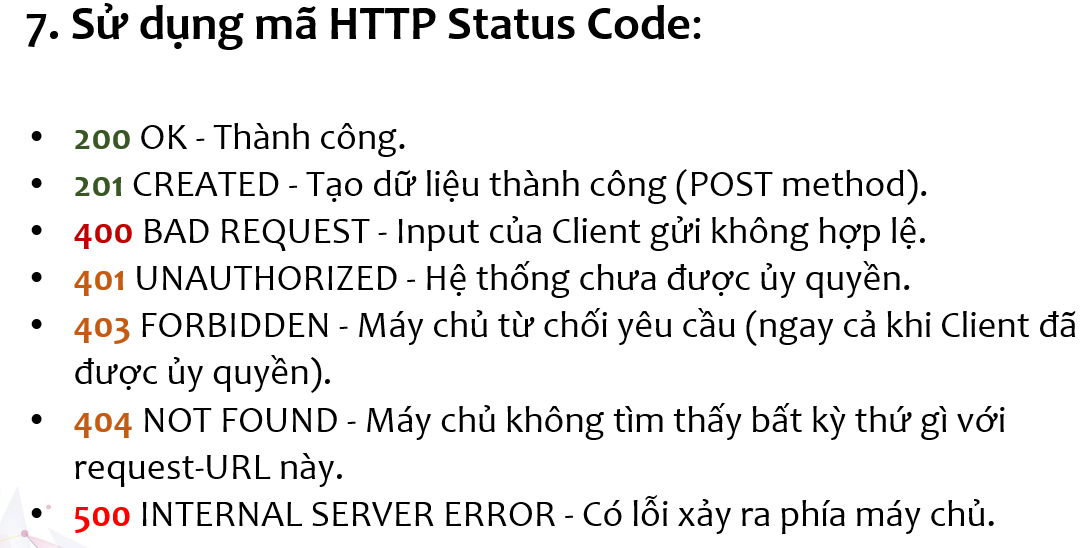
Dataset là đối tượng dùng để chứa nhiều Datatable với nhau và các mối quan hệ giữa chúng, còn datatable dùng để chứa dữ liệu, có tên, cột và các dòng, ánh xạ của một bảng trong csdl

Datatable giống như những chiếc hộp chứa dữ liệu còn dataset giống như chiếc thùng to để đựng những chiếc hộp đó

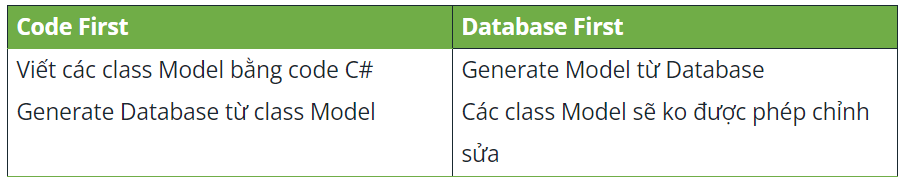
* **RESFULL API, khái niệm session, cookie, token**
* RESful API không phải là một loại công nghệ.
* RESful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng Web (Web Service).

Hoạt động chủ yếu dựa vào giao thức HTTP (dùng chủ yếu 4 loại get, post, put, delete)

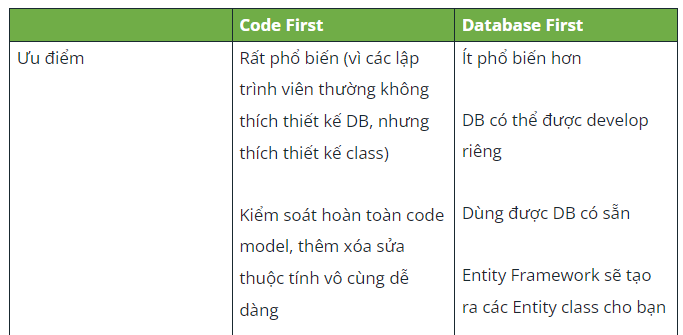
RESTful API không sử dụng Session và Cookie. Nó sử dụng một access\_token với mỗi request (session là phiên làm việc được lưu trữ trên sever; cookie là phiên làm việc lưu trữ dưới client; access\_token: mỗi một request gửi lên, nó sẽ kèm theo một token (một mã) được sinh ra gửi lên sever, ta sẽ phải sử lý dữ liệu về cái token này để phân tích để xem cái thông tin này là muốn xử lý vào một thông tin gì đó)



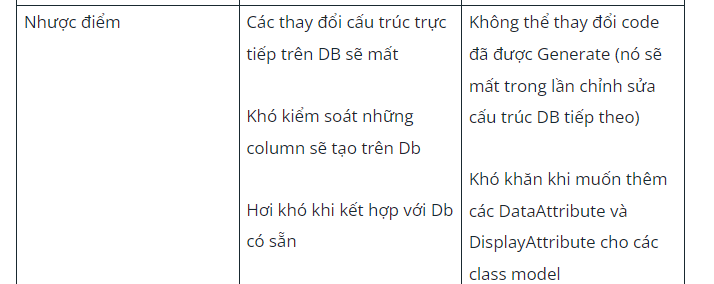
* **Code first và DB first**
* Giải thích



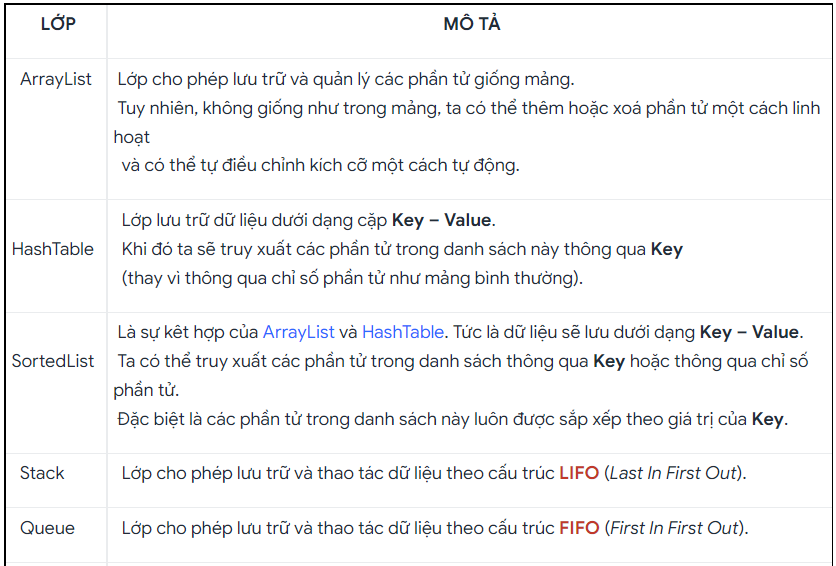
* Ưu điểm



* Nhược điểm



* **Collection là gì?**
* Là một mảng có kích thước động:
* Không cần khai báo kích thước khi khởi tạo
* Có thể tăng giảm số lượng phần tử trong mảng 1 cách linh động
* Một số lớp conllection được sử dụng phổ biến:

****

* **Generic là gì?**

Cho phép định nghĩa 1 hàm , 1 lớp mà ko cần chỉ ra đối số kiểu dữ liệu gì. Tùy vào kiểu dữ liệu mà người dùng truyền vào thì nó sẽ hoạt động theo kiểu dữ liệu đó.

**public** **static** **void** Swap<T>(**ref** T a)

{

T temp = a;

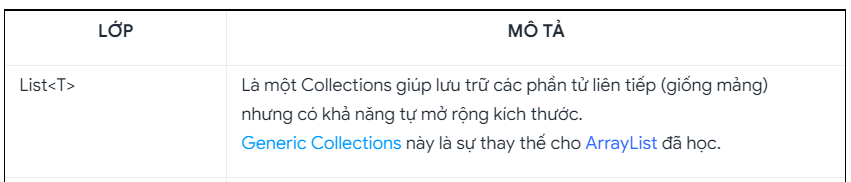
Return temp;

}

* **Generic Collection?**

Là các mảng động có thể linh động trong việc truyền giá trị có kiểu dữ liệu khác nhau vào.

Generic Conllection phổ biến:



* **Delegate trong C#**
* Delegate là một biến kiểu tham chiếu chứa tham chiếu tới một phương thức
* Bạn cứ hiểu Delegate là một**biến bình thường**, biến này chứa hàm mà bạn cần gọi. Sau này lôi ra sài như hàm bình thường. Giá trị của biến Delegate lúc này là tham chiếu đến hàm. Có thể thay đổi runtime khi chương trình đang chạy.
* **Từ khóa static**

Sinh ra nhằm giải quyết mong muốn 1 thuộc tính nào đó có thể dùng chung cho mọi đối tượng (chỉ được cấp phát 1 vùng nhớ duy nhất, được khởi tạo ngay khi chạy chương trình và kết thúc khi đóng chương trình)

Có 4 loại:

* Biến tĩnh: public static int BY= 10;
* Phương thức tĩnh: public static void A()

{

Return 10;

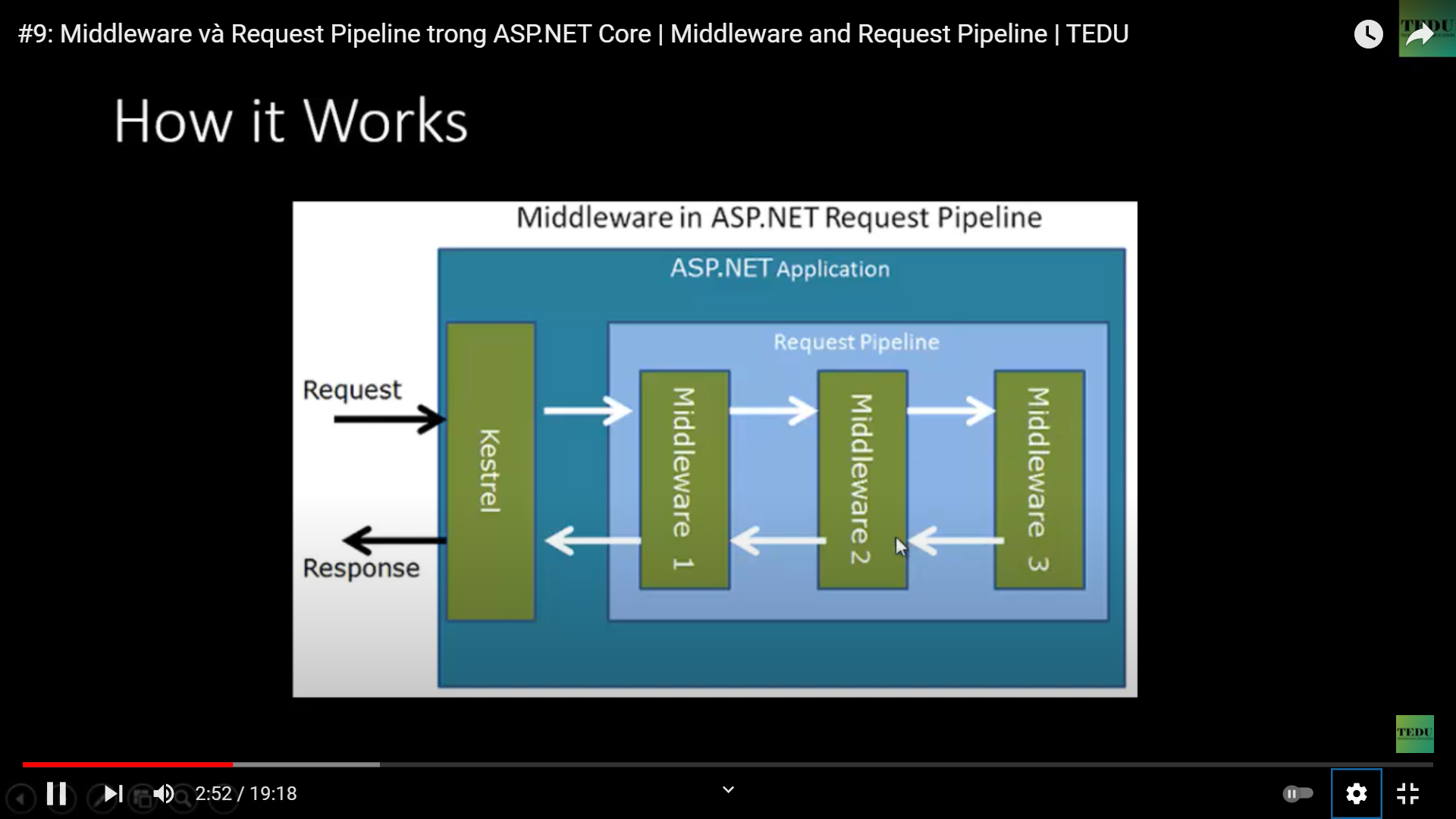
}

* Lớp tĩnh: public static class BY{}
* Phương thức khởi tạo tĩnh: …

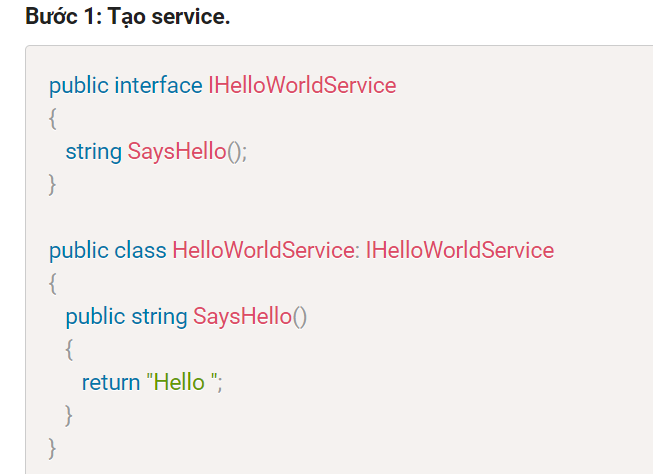
# Request Pipeline là cơ chế bắt đầu khi các request bắt đầu được xử lý với một đối tượng Request đầu vào và kết thúc với đầu ra là một response

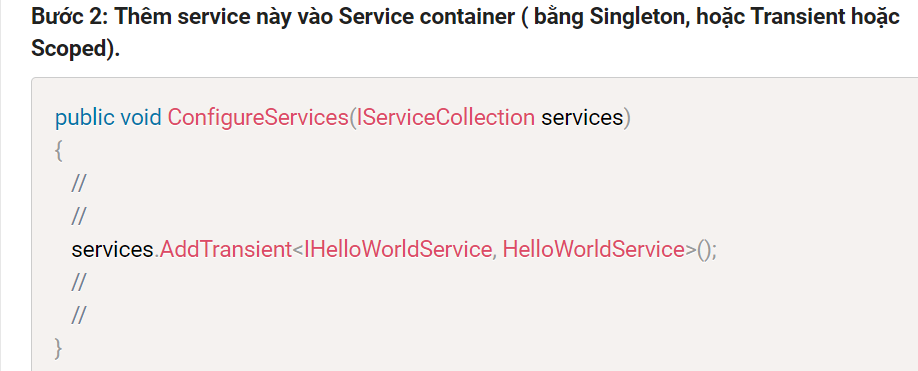
* **Middleware**

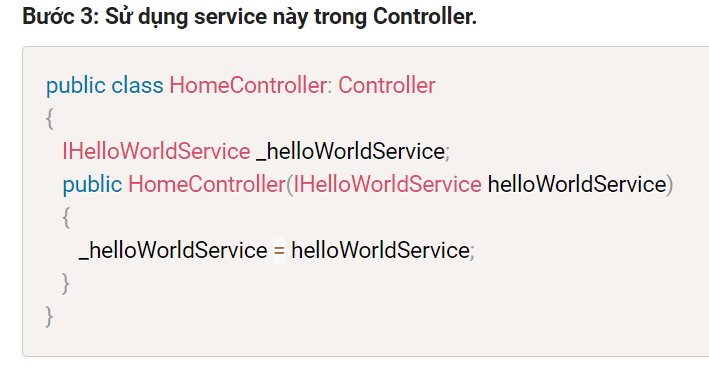
Middleware là thành phần ở giữa can thiệp vào request pipeline (luồng request) để xử lý chúng và tạo ra response phản hồi lại client.



**Làm sao để thêm một Service Dependency vào Controller?**







~ public void ConfigureServices(IServiceCollection services): dùng để cấu hình các sevice được sử dụng trong ứng dụng. Hàm này được gọi đầu tiên khi ứng dụng được request lần đầu

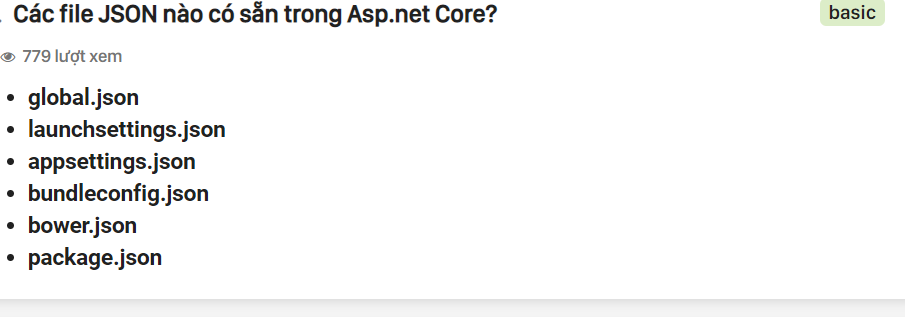
~ public void Configure(IApplicationBuilder app, IWebHostEnvironment env) : Nó định nghĩa cách mà ứng dụng sẽ hồi đáp tới mỗi HTTP request. Sử dụng hàm này, chúng ta có thể cấu hình Middleware có sẵn như Routing, authentication, session,cors …

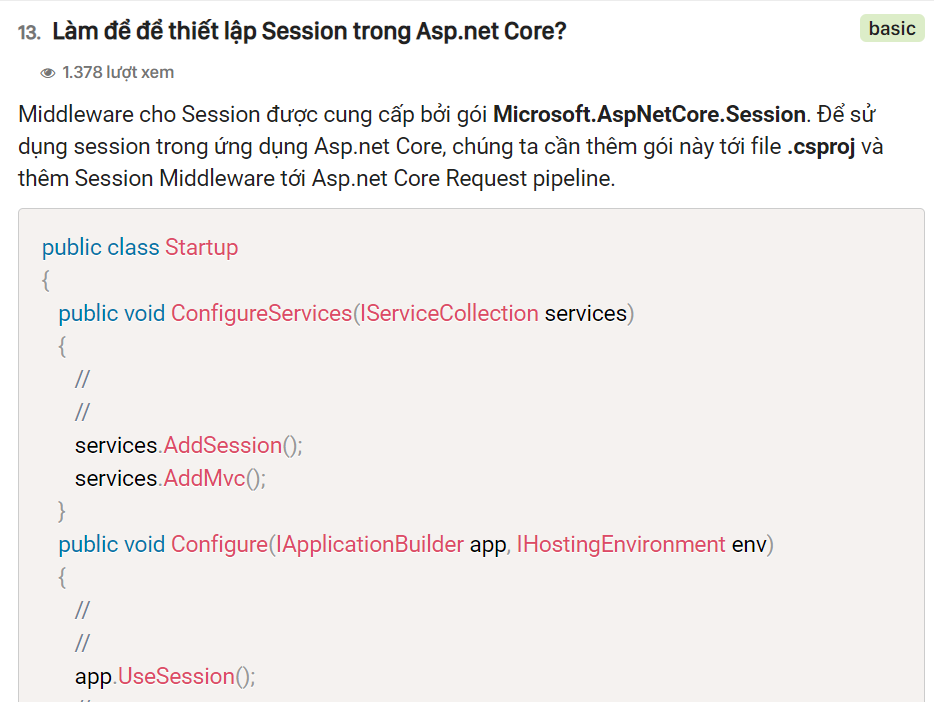
**Serialization là gì?**

- Là sự chuyển chuyển đối tượng thành 1 luồng các bytes.

**Sự khác nhau giữa Property và Field?:**

**Property** dùng để quản lý sự truy cập đến **Field**. Dùng **Property** để đảm bảo **Field** không bị gán các giá trị không hợp lệ.





JWT (JSON Web Token) là một chuỗi mã hoá chứa dữ liệu ,được dùng để truyền tải thông tin giữa các hệ thống dưới dạng json một cách an toàn .

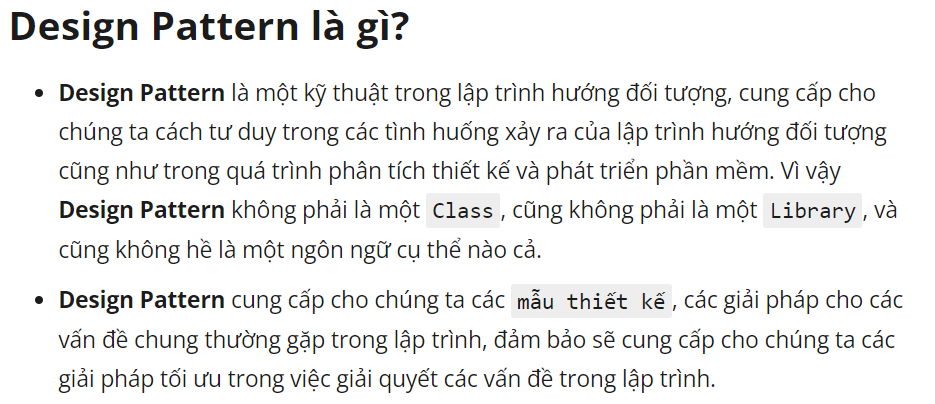
### **3 thành phần của JWT:**

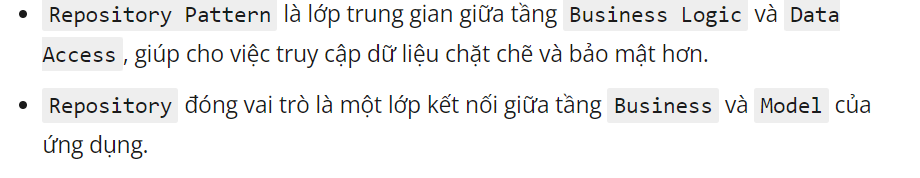
Header: bao gồm hai phần chính: loại token  và thuật toán đã dùng để mã hóa

Payload chứa các claims. Claims là một các biểu thức về một thực thể (chẳng hạn user) và một số metadata phụ trợ. Có 3 loại claims thường gặp trong Payload: reserved, public và private claims.

Signature trong JWT là một chuỗi được mã hóa bởi header, payload

Docker: để build dự án trên đa nền tàng





Công ty Cổ Phần Công nghệ NexData được thành lập năm 2011. Với 10 năm kinh nghiệm hoạt động trong lĩnh vực công nghệ thông tin, công ty là nhà cung cấp sản phẩm, giải pháp, dịch vụ công nghệ hàng đầu tại Việt Nam

Sứ mệnh:  
Nghiên cứu, cập nhất, phát triển các sản phẩm, giải pháp, dịch vụ công nghệ số hiện đại nhằm mang đến sự hài lòng cao nhất cho khách hàng, đồng hành cùng Chính phủ và doanh nghiệp trên hành trình chuyển đổi số, xây dựng nền kinh tế số – xã hội số.

#### TẦM NHÌN CHIẾN LƯỢC

Trở thành công ty công nghệ hàng đầu với năng lực, kinh nghiệm, uy tín, và chất lượng dịch vụ chuyên nghiệp, góp phần nâng cao vị thế của nền công nghệ thông tin Việt Nam trên bản đồ thế giới.