Office Tools

Thursday, November 23, 2023 11:19 AM

Notes

Thursday, November 23, 2023 11:21 AM

Assignments

Thursday, November 23, 2023 11:21 AM

SI.No	Excercise	Assignments Files

Issues And Solutions

Thursday, November 23, 2023 11:27 AM

Sl.No	Issues	Solutions

SQL

Thursday, February 8, 2024 12:08 PM

Structured Data => Perfectly fit into table

Unstructured Data ==> Socila Media

Semi structured Data ==> Structured + Unstructured
Email ==> header send , To + Unstructured Content Page

Relational DataBase :

Structured and in the form of Table

SQL

NON -RELATIONAL DATABASE :

NOT IN THE FORM OF TABLE .IT IS CONTAINED A UNSTRUCTURED DATAS

NO SQL (NO USING OF SQL) . GRAPH TREE COLUMN WISE IN ANY FORM

Schemas

Thursday, February 8, 2024 12:20 PM

In the world of computers, when people store information in a database (which is like a huge digital box for keeping data), they use something called a "schema" to organize this information. A schema is like the plan or map for where everything goes in the database. It tells the computer:

DDL(Data Definition Languagew)

Thursday, February 8, 2024 12:22 PM

1. CREATE

The CREATE command is used to create a new database, table, index, or other database objects.

```
CREATE TABLE Students (
   StudentID int,
   StudentName varchar(255),
   Grade int
);
```

2. ALTER

The ALTER command is used to modify an existing database object, such as adding, deleting, or modifying columns in an existing table. For example, to add a new column to the Students table:

```
ALTER TABLE Students ADD Email varchar(255);
```

3. DROP

The DROP command is used to delete an existing database, table, index, or view. This command removes the object and its data permanently. For example, to delete the Students table:

```
DROP TABLE Students;
```

4. TRUNCATE

The TRUNCATE command is used to delete all the rows from a table without deleting the table itself. This can be much faster than deleting rows one by one. For example:

```
TRUNCATE TABLE Students;
```

5. COMMENT

The COMMENT command is used to add comments to the data dictionary, which is not universally supported in all database systems but can be found in some. It's used for documentation purposes within the database schema.

6. RENAME

The RENAME command is used to rename an existing object in the database, such as a table. The exact syntax for this command can vary between different database systems.

DML (Data Manipulation Language)

Thursday, February 8, 2024 12:25 PM

DML stands for Data Manipulation Language, which is a subset of SQL (Structured Query Language) used to manage and manipulate data within database objects like tables. Unlike Data Definition Language (DDL) commands, which define and modify the structure of database objects, DML commands are used to read, insert, update, and delete data in the existing objects. Here are the main DML commands:

1. SELECT

The SELECT command is used to query data from a database. It allows you to retrieve data from one or more tables, with various options to filter, group, and sort the data according to your needs. For example:

```
SELECT StudentName, Grade FROM Students WHERE Grade > 75;
```

2. INSERT

The INSERT command is used to add new rows of data to a table. You can insert values into specific columns or into all columns of a table. For example:

```
INSERT INTO Students (StudentID, StudentName, Grade) VALUES (1, 'John Doe', 88);
```

3. UPDATE

The UPDATE command is used to modify existing data in a table. It allows you to change values of the specified columns in rows that match a particular condition. For example:

```
UPDATE Students SET Grade = 90 WHERE StudentID = 1;
```

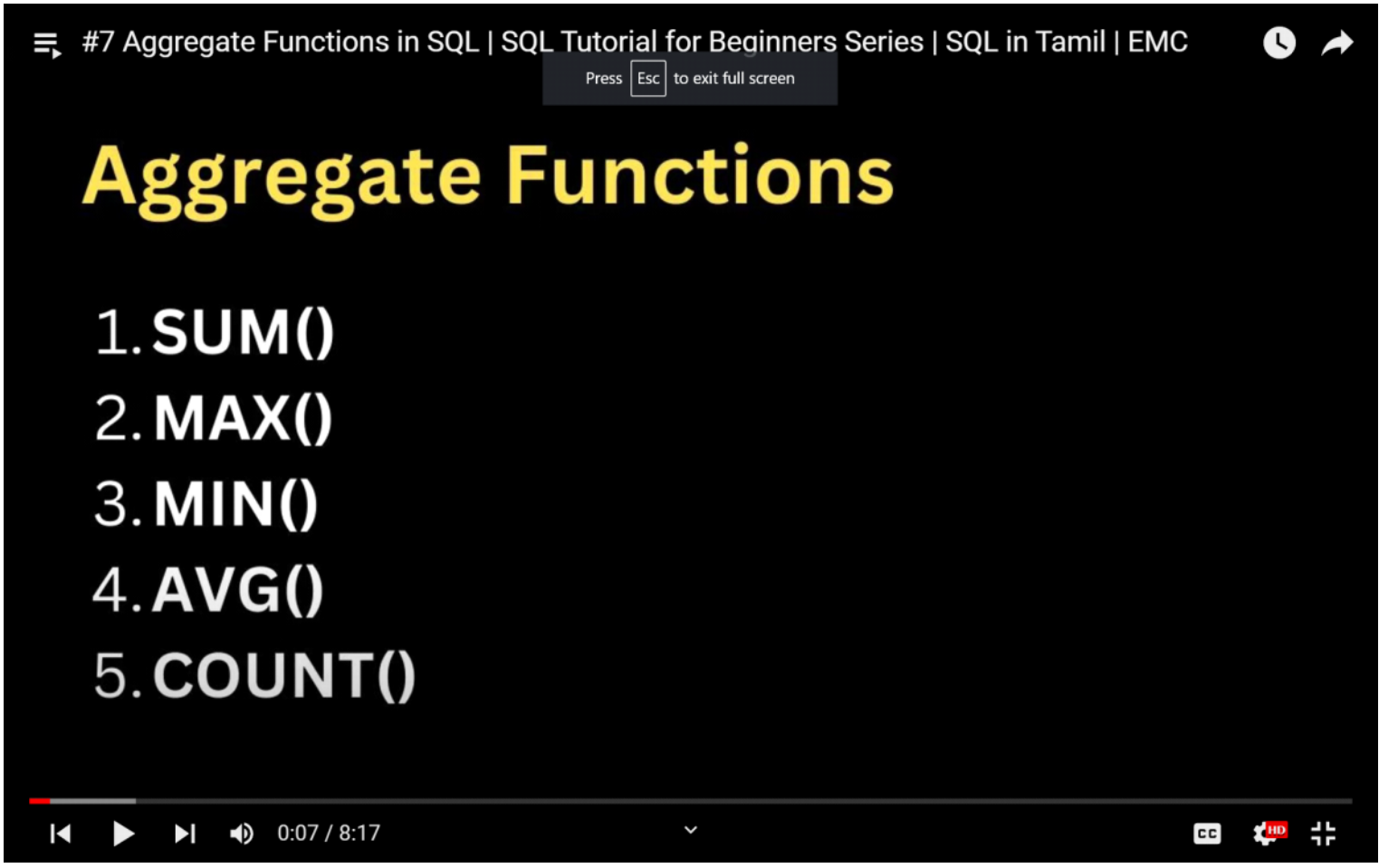
4. DELETE

The DELETE command is used to remove rows from a table based on a specific condition. If no condition is provided, all rows in the table will be deleted (which should be done with caution). For example:

DELETE FROM Students WHERE Grade < 60;

Aggregate Functions

Thursday, February 8, 2024 12:31 PM



Group By Order By

Thursday, February 8, 2024 12:32 PM

GROUP BY Clause :

The GROUP BY clause groups rows that have the same values in specified columns into summary rows, like "find the number of customers in each country." It is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()), which perform a calculation on a set of values and return a single value.

SELECT Country, COUNT(CustomerID) AS NumberOfCustomers FROM Customers GROUP BY Country;

ORDER BY Clause :

The ORDER BY clause is used to sort the result set returned by a SELECT statement in ascending or descending order, based on one or more columns. By default, ORDER BY sorts the data in ascending order. To sort the data in descending order, you can use the DESC keyword.

SELECT CustomerName, Country FROM Customers ORDER BY Country ASC, CustomerName DESC;

Using GROUP BY and ORDER BY Together

You can use both GROUP BY and ORDER BY in a single query to group your data and then sort the grouped data.

SELECT Country, COUNT(CustomerID) AS NumberOfCustomers FROM Customers GROUP BY Country ORDER BY NumberOfCustomers DESC;

Having By

Thursday, February 8, 2024 12:35 PM

Imagine you have a table named Sales with columns for SalesPerson, SaleAmount, and SaleDate. You want to find out which salespeople have achieved more than $10,000 in total sales. You would use the GROUP BY clause to aggregate sales by salesperson and the HAVING clause to filter the results:

```
SELECT SalesPerson, SUM(SaleAmount) AS TotalSales
FROM Sales
GROUP BY SalesPerson
HAVING SUM(SaleAmount) > 10000;
```

C#

Thursday, November 23, 2023 11:19 AM

Vino Krishnan: https://learning.syncfusion.com/?sfwd-courses=c-fundamentals-for-ab...
sent on December 27, 2023 3:45 PM
C# Fundamentals Course Link

LinQC#	LINQ - Tamil (Part – 1 - Introduction) 	 PracticeLinQ	From Where Join Having Select

Notes

Thursday, November 23, 2023 11:21 AM

Create a new folder without space then open new terminal inform that

Purpose	**cheat code**	**Output**
for visual studio open in terminal	code .	opened
Open with that folder	dotnet new console	Restore succeeded.
For output	Dotnut run	

Assignments

Thursday, November 23, 2023 11:21 AM

SI.No	Excercise	Assignments Files
1.	C# Switch statement From <https://traininghub.syncfusion.com/myassignments>	 Switch_statement
2 .	C# For Loop From <https://traininghub.syncfusion.com/myassignments>	 for_loop
3.	C# While Loop From <https://traininghub.syncfusion.com/myassignments>	 While Loop
4.	C# Array From <https://traininghub.syncfusion.com/myassignments>	 array_classroom_assi...
5.	Do While Loop - Classroom From <https://traininghub.syncfusion.com/myassignments>	 do_while
6.	C# Methods - Classroom Assignment From <https://traininghub.syncfusion.com/myassignments>	 Methods
7.	# String Manipulation	

	From <https://traininghub.syncfusion.com/myassignments>	
8.	# String Manipulation From <https://traininghub.syncfusion.com/myassignments>	

Issues And Solutions

Thursday, November 23, 2023 11:27 AM

Sl.No	Issues	Solutions
	1/2 la output 0.5 nu venum nda atleast oru number aaachum double ah irukkanum	1/2 la output 0.5 nu venum nda atleast oru number aaachum double ah irukkanum
	Case la string case la single quote pottathunaala work aagala	String la epavume double quotes thaan podanum

	<pre>case "+" :
	int ans=n1+n2;
	break;
	case "-":
	ans=n1-n2;
	break;
	case "*":
	int ans=n1*n2;
	break;
	case "/":
	int ans=n1/n2;
	break;
	case "%":
	int ans=n1%n2;
	break;</pre>
		Int one time assign pannitu again assign panna koodaathu

	{
	
	}
	Scope kulla irukura visayam ah velila access panna mudiyathu…
	Already initialize pannuna visayatha again initialize panna thevai illaaai..kooodathu	

	Date and Time la `DateTime dob=new DateTime(2016,8,16,3,57,0);` Output: 08/16/2016/ 03:57 If I give `DateTime(2016,8,16,3,57,00);` Output: 08/16/2016/ 03:57:00	
	DATEtIME la new keyword pottu eluthalam like new datetime or datetime.parseExact podalam	
	Exception because of not using after login page creation..and also I used flag so it will again changes so that we must break to reduce the code and code flag again running with false /true confusions Break;	

OOP

Thursday, November 23, 2023 11:19 AM

NOTES

Thursday, January 4, 2024 11:48 AM

OOPS :

Difference between procedural oriented Language and Object Oriented Language
OOP is faster and easier to execute
OOP provides a clear structure for the programs
OOP makes it possible to create full reusable applications with less code and shorter development time

CLASSES :

Classes are the Blueprint or Template which we can use for creating objects .

Objects :

Objects are instantiated from the Class . From Creating One class we can create many objects in the classes .

Class Members :

Class have members of feields , methods ,properties ,events .

Constructors :

Constructors are the special method to initialize the objects .(Definition for class .Initialize the class)

Default Constructror:

If we didn't give anything Default Consatructor will be activated .

Parameterized Constructor :

To initialize field and access properties to get and set .

Access Modifiers :

Public -Public for all Access for everything
Private-Within the class
Protected -That Class +That Inherited Class
Internal-That Project Only of Namespace

Properties :

Properties is that kind of variable + Methods By using Get and Set .

Inheritence :

Base class and Derived Class
Derived class(Child Class)
Access Specifiers Base Class(Parent Class) : Derived Class(Child Class)
Through that Inheritence , We can save a lot of times and it is a very importace to Code Reusabillity .
The Main Factor is we didn't need to write again and again to that code of base class .

Sealed Class :

In case we use sealed class , we can not to be inherited to that derived class .Sealed class is for protection .

Polymorphism :

Method Overloading and Method Overriding .

Method Overloading :

Method name is same but Constructors are changing .
For that situation, we can use method Overloading . If the user gives double parameters then it will call the function of double that p[articular variable parameters .

Method Overriding :

Method Name is same method name even same parameters so that we can give it to that .
Through the Keywords of **Virtual and Override** we can attain Method Overriding .

Abstraction :

Hiding the essential Things and showing only the necessary details of the class .

Abstract keyword is used and through inheritence it can be utilized properly when it will be used .

Interface :

Interfaces are the empty bodies . They don't have the proper definition in that. We can inherit and use it . Blueprint for that .

Enum :

Enum is a special class which is representing unchangebel Constant things .

PostgreSQL

Thursday, January 4, 2024 2:13 PM

Notes :

Thursday, January 4, 2024 2:14 PM

- CREATE keyword uses to create the table
- After create the table We can implement the fields into the tables which will be the heading of the table
- By using Select * table keyword we can display the whole table .
- VARCHAR is for string and also if didn't know the size we can just give 255.
- INSERT INTO :
  Keyword used to insert values into that table in the order of fields which we created before .
- Insert Multiple Rows :
  We can give multiple values with the rows coma separated .
- Select Particular things like brand , year from the car database .
- ALTER Table -keyword :
  We can create another column into that table .
  By using add keyword we can create heading for that new column name .
- Update Statement :
  By using the update statement , we can update the values into already present into that table
  Also by using the where keyword we can specifically choose which is going to be change
- Where Keyword :
  We can specify which is like that
- Update Multiple Columns :
  We can update multiple columns by using comma .The Keyword "SET" is used to set the value .
- Alter the data types in the values
  Altering the data types which is into the table for example int year already there so we can easily changing into that varchar year(4); like that
- Alter the size of the color which is 255 --->30 like that
- DROP Column :
  Drop column is used for to remove the column in that table by using the "DROP " keyword .
- Delete Keyword :
  Delete keyword is used to delete all of the things by which we are mentioning in that table by giving that particular or full code of things .
- Truncate table used to delete all the records in that data base .
- Operators in where clause :
  Greater than>
  Lesser than<
  = equals to
  Is null
  Greater than equals to>=
  Lesser than equals to<=
  Not equals to <> !=
  Like
  I like
  And
  Or
  In
  Between
  Is null
  Not(Not IN, Not Between ,Not Like)
- Select Data :
  Select Data is used for selecting for particular data or whole data.

- Select distinct :
  Select distinct means remove the duplicate elements .
- Select distinct count :
  Select distinct count means count the number of different elements in the distinct .
- Filter Records :
  By using filter keyword we can easily in the form of "WHERE" keyword .
- Sort Data :
  For sorting data , we can use order by keyword is used .
- Sort Data:
  For sorting data in the form of descending order , we can use "DESC" keyword is used .
- Limit clause :
  Limit clause is the thing in the form of limiting
- Limit -OFFSET :
  Limit Offset is the thing we can use it to the offset …offset keyword is for beginning index ..from where it may be start like that .
- MIN:
  Min defines the minimum values in that table .
- MAX :
  Max defines the maximum values in that table .
- AS- Keyword :
  AS Keyword refers to create a new name for the value .
- Count Function:
  Count function refers to the count of sum
- SUM:
  Sum Function refers to the total for that field which we are given .
- AVG Function :
  Avg Function returns to the avg for the field of values .
- Avg Function:
  With Two decimals :numeric(10,2)
- LIKE(Case Sensitive) :
  We can use it like starting f letter in the database
  Ending letter
  Middle Letter in that Database .
  Underscore is used for like used of fill in the blanks like that use of case .
- I LIKE(Non Case Sensitive) :
  We can use it like starting f letter in the database
  Ending letter
  Middle Letter in that Database .
  Underscore symbol used for like fill in the blanks of cases.
- IN
  IN =====> similar like that "OR" .
- Not IN
  NOT IN ===> like that not in of(OR)
- Between Operator
  Used to select the values in between the two values from that .
- Concatenate Columns :
  By Using the keyword of || and also give string and space for our requirement by using "AS" keyword we can create a separate list for that .
- Inner Join :
  Show Only exactly perfect match part only .
- Left Join :
  Show Matched and also in the left first table which is primarily .
- Right Join;

Shows the matched items and also in the right second table which is in the top

- Cross Join:

  Cross join matches with every other table content .

- Group :

  Group is for grouping the datas .

- Union :

  Union is for union the same data types of two or more tables .But it removes the duplicates of elements .

- Union ALL :

  It doesn't remove duplicate all the elements .

- Having

  In the functions like Group By we cannot use the where..so alternative is having .

Assignments

Friday, January 5, 2024 3:40 PM

Topic	File
SELECT	 SELECT
Column Alias	 Column Alias(Con...
Distinct	 Distinct
Order BY	 OrderBY
Order BY Via Nulls	 Nulls first last asc d...
Distinct	 Distinct
Limit	 Limit

Fetch	 Fetch
IN	 IN Operator
Between	 Between Operator
Like	 Like ILike Pattern M...
Null	 Not Null
8/1/2024	 inner Join  Left Join  Right Join  Full Outer Join

Full Outer Join

cross Join

natural Join

Having Group By

Having Group By 2

Union Example

Intersect Table

Except Operator

Boolean DataType

Character

Character DataType

Numeric DataType

Integer DataType

CurrentDate

Date DataType

Timestamp

Time DataType

UUID(Universal Uniqu...

Array

	 Json DataType  Any All Exists  Unique , Check, No...  case
9/1/2024	 case  Sequence  Identity  Alter Table  rename table

Add Column

drop column

Column Type Cha...

Column Rename

Drop Table

Truncate Table

Temp Table

Dot Net Core MVC

Wednesday, January 10, 2024 11:07 AM

Dependency Injection :

Loosely Coupled
Worked in a single place not need to work on every page
Flexibility

- Constructor Dependency Injection.
- Property Dependency Injection.
- Method Dependency Injection.

Create the project :

==>project name, solution name must be different .

Launch Settings :

==>Launchsettings.json in that there is the url amd port numbers .
=> In the root files , Static files , no c# files and html, css ,js files are can be added .
=>Folders for models views controllers .
=>AppSettings. Json-Connection strings and secrets of details are there present in there.

Program.Cs :

Create Container
Add services to the Containers
Configurations
Middlewares
mapControllerRoute(
Name:"default"
Pattern:"{controller=Home}/{action=Index}/id}";
In this Order of pipelines are very important .Like First Authentication then Authorization .
)

MVC Architecture :

Routing In MVC :

URL/{controller}/{action}/{id}

Controller :

Controller can have multiple actions .
Controller default base class .

Views :

Home -Default
Shared :
1.Layout->Masterpage
2.Validations->ScriptsPage

3.Error
4.ViewImport
5.ViewStart->StartingPoint ==>Layout=_Layout;

TagHelpers :

Special Tags looks like HTML
Asp-Controller and asp-action
C# code into Html Code .

Action Result :

IActionResult ->actions of common methods/pages

EntityFrameWorkCore :

Createb DB and connect Implementation .

AttributeRouting

[Route("Raja")]

Data Annotations :

[Key]

Views :

.csHtml files
1.Model=
Return view(model);
2.AbsolutePath :
Return view("MyViews/Test.cshtml");
3.RelativePath:
Return view("");

View Data :

<h3>@ViewData ["Page Title"]</h3>
@{
//ViewData Dictionary
Var employee=viewData["Employee"]

Name : @employee.Name

View Bag :

RunTime Checking
<h3>@ViewBag.pageTitle</h3>

Strongly Typed Views :

Return view(model);

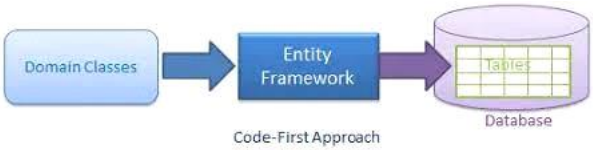
.csHtml File
@model Directive

Entity FrameWork Core :

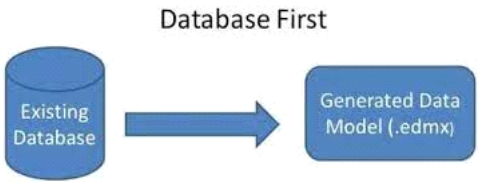
Object Relational Mapper :(Bridge between Database and OOP)

1.Code First Approach
2.Db First Approach

Code First Approach :



DataBase First Approach :



DataBase Providers :

EF Core => Providers => DataBase

EF Core Packages :

.sqlServer Install Only ==>.Realtional==>.Core

Installing automatically .

DbContext :

```
Public class AppDbContext : DbContext{

    Public AppDbContext(DbContextOptions <AppDbContext> options ) : base(options)
    {

    }

    Public DbSet<Employee> Employees{get;set;}
    //Here we can add the initial Data Seeding funtions with model Builders

}
```

Connection String :

```
 "ConnectionStrings": {
   "DefaultConnection": "Host=34.23.55.201;Database=training-db;Username=freshers-
training;Password=kV1O1st2Uukp9=2"
 },
Trusted_Connection =true;
Integrated_Security=SSPI;
Integrated_Security=True ;
```

Repository Pattern :

IEmployeeRepository
->Create
->Read
->Update
->Delete

Here from this InterFace we are creating In Memory Repository and SQL Repository .

Migrations :

Add-Migrations
==>Migrations files Created
Update DataBase
==>in that DataBase datas will be updated .

Seeding Initial Data :

In the Application DbContext , we can seed the initial Datas in the DbContext Page .

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
   //base.OnModelCreating(modelBuilder);
   modelBuilder.Entity<Books>().HasData(
      new Books(1, "Rich dad Poor Dad", "Robert kiozakie", 1200),
      new Books(2,"Edge of Tommorrow","Steven Speilberg",1300),
      new Books(3,"Atomic Habits","James Atomic",1400)
      ); ;
}
```

Using Services sql server with EF Core :

Program.cs la ADDIng DB:

```
builder.Services.AddDbContext<BookLibraryContext>(options =>
{ options.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection")); });
```

ModelBinding :

Model Binding is referred to as the information about the mapping to the object .

```
public IActionResult Edit( Players player)
```

Here In the form submit will be happened and the details of the players from the players class will be automatically binded here So It is called as Model Binding .

Fluent Validation

Friday, February 2, 2024 2:33 PM

Fluent Validation:

Explore Fluent Validation as an alternative to Data Annotations for complex validation rules. Learn how to configure validation rules using a fluent API in a separate class. Model Binding and Validation: Gain insights into how model binding works in ASP.NET Core MVC. Learn to customize model binding and handle model binding errors effectively. explain to a small child

AJAX

Friday, February 2, 2024 10:25 AM

Imagine you're playing with your favorite toy, and you decide you want to play with another toy without putting the first one away. AJAX is like magic that lets websites get new toys (or information) without having to put the old ones away first or start over from the beginning.
Let's say you're drawing on a piece of paper, and you want to ask your friend what color you should use next without stopping your drawing. AJAX is like asking your friend, and they whisper the color back, so you can keep drawing without stopping or looking up. It makes things faster and smoother, just like magic whispers!

Let's imagine you're playing a game on a tablet where you're building a zoo. You have animals, trees, and other items to place in your zoo. Every time you add something new, like a lion or a tree, you expect it to show up immediately without having to restart or refresh the game. This is very similar to how AJAX works on websites.
For a real-life example, think about when you're using a website like Google Docs to write a story or do homework with your friends. As you type a sentence, your friend can see what you're writing instantly, even if they are far away in their house. You don't have to save, close the document, and send it to them every time you add something new. Your changes appear on their screen as if by magic!

This happens because of AJAX. It's like a little helper that runs back and forth between your computer and the internet, bringing back the new things you or your friends add to the document without having to reload the whole page. It makes using the website smoother and lets you see changes in real time, just like adding animals to your zoo in the game without any interruptions!

ACCEPT VERBS

Friday, February 2, 2024 10:08 AM

[httpget]
[httpPost]

For Giving in the same page , we can give

Like

AcceptVerbs ("Get","Post")

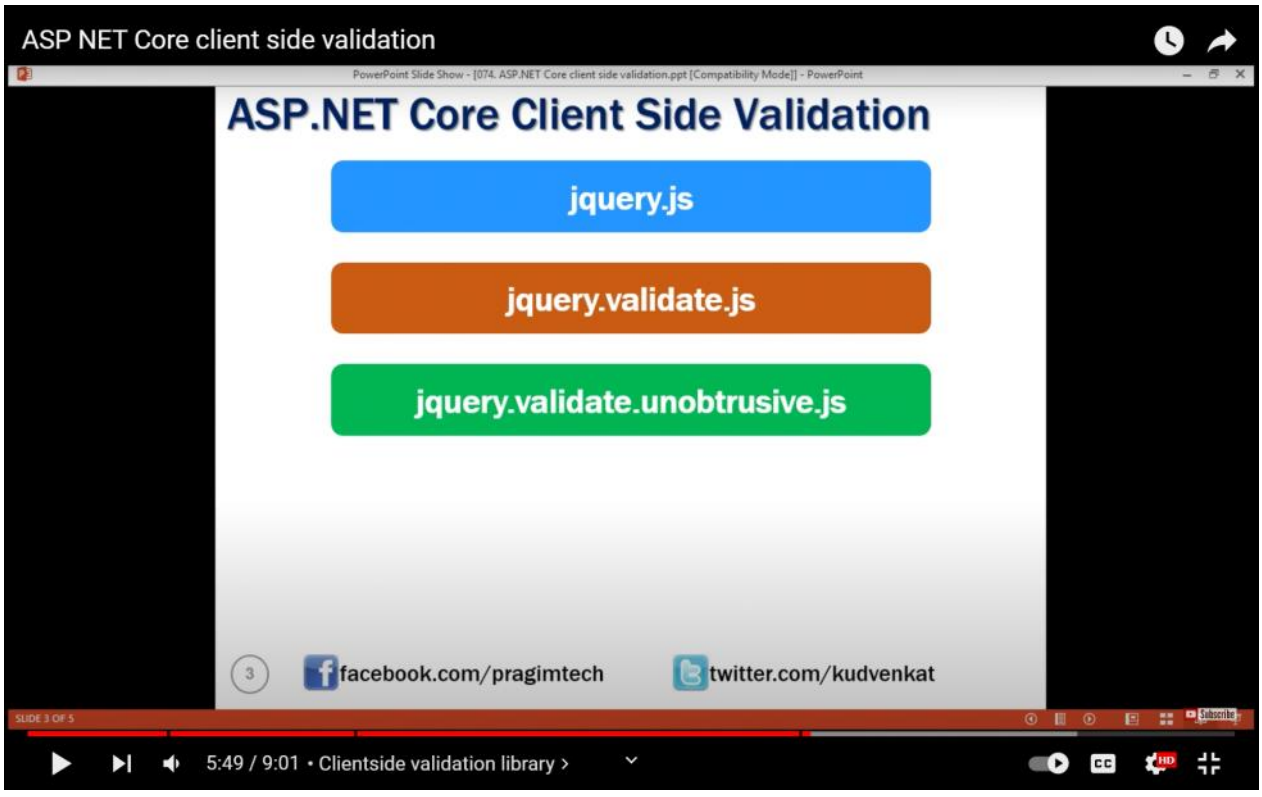
REMOTE VALIDATIONS

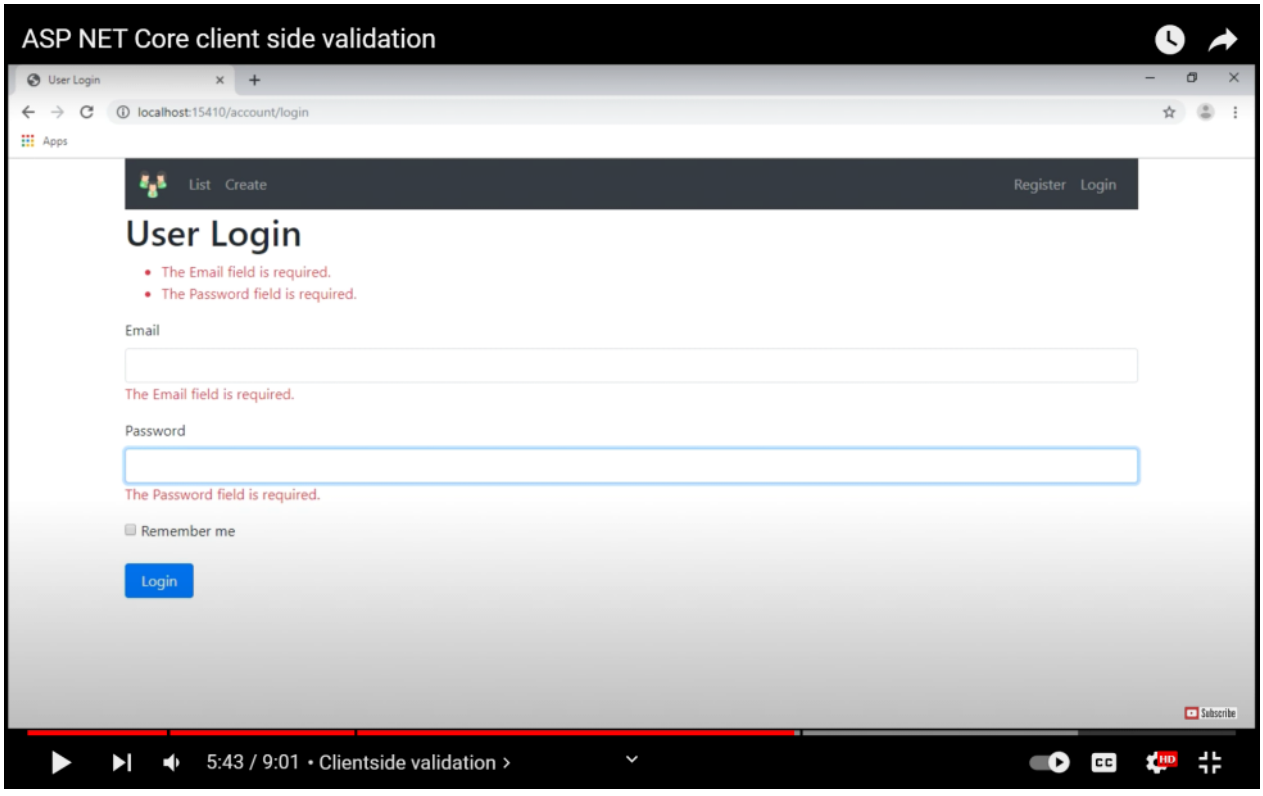
Friday, February 2, 2024 10:07 AM

Remote validation in ASP.NET Core MVC is a powerful feature that allows you to perform client-side validation asynchronously by making an AJAX call to a server-side method. This is particularly useful for validations that cannot be determined on the client-side alone, such as checking the uniqueness of a username or email address in a database.

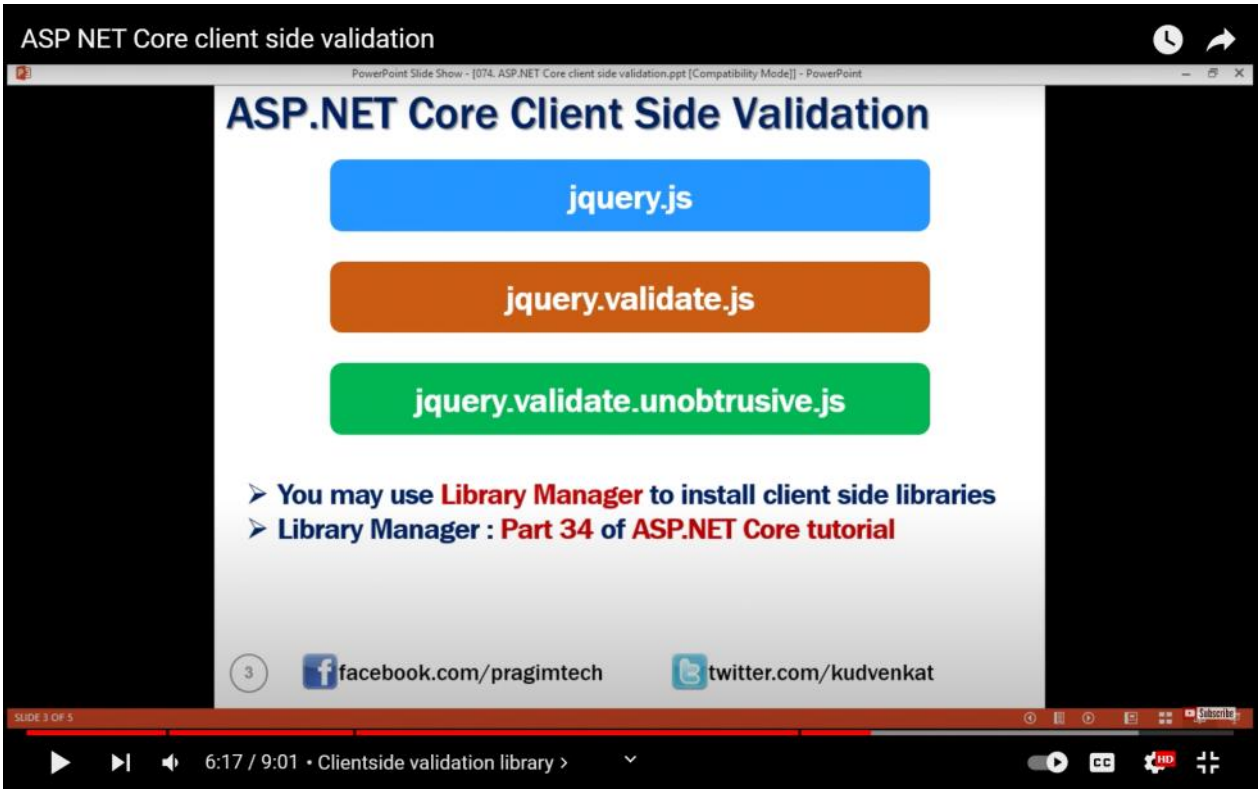
Client Side Validations

Thursday, February 1, 2024 2:40 PM





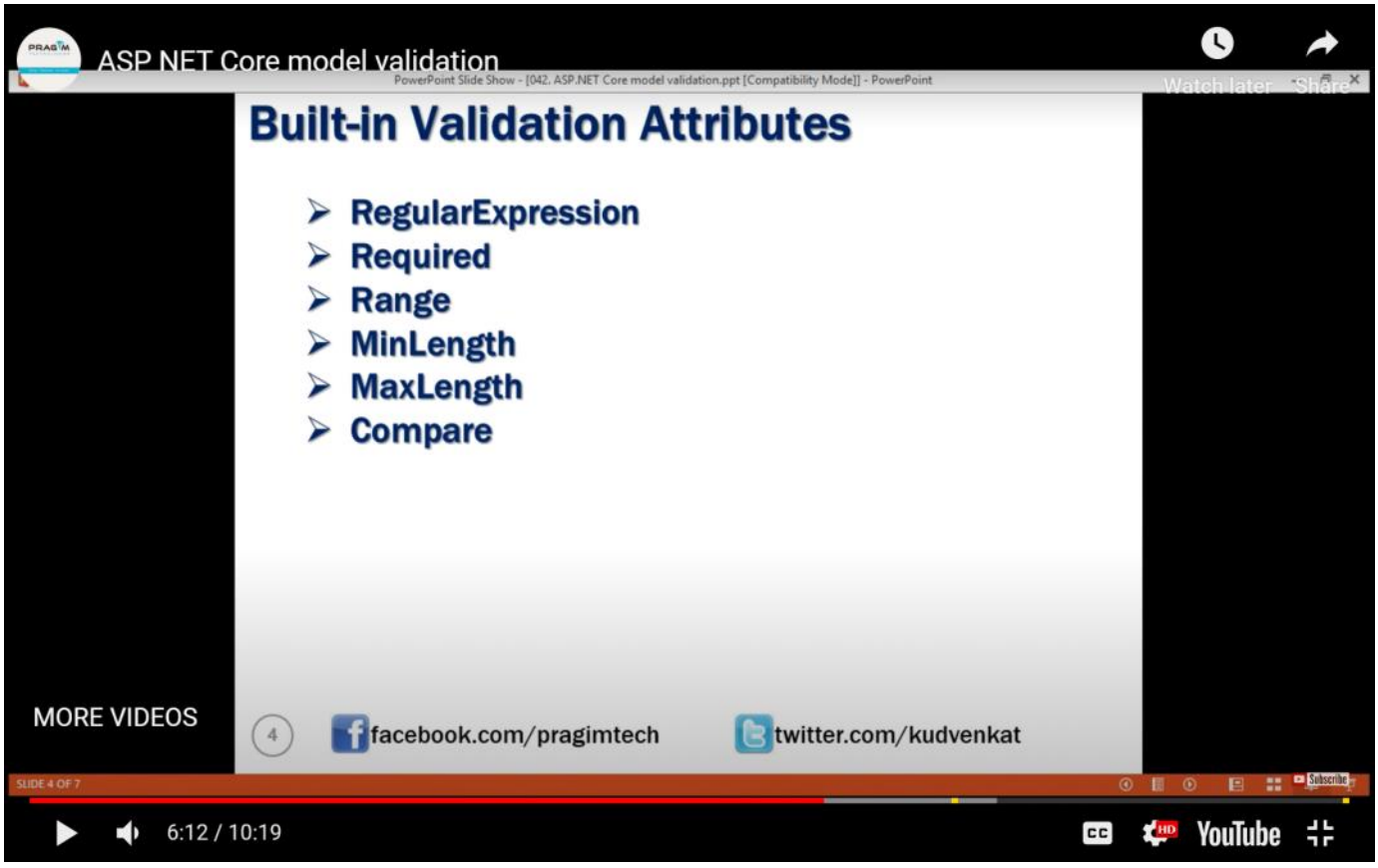
The load of the server is reduced
And also the client side ui so, it is very fast

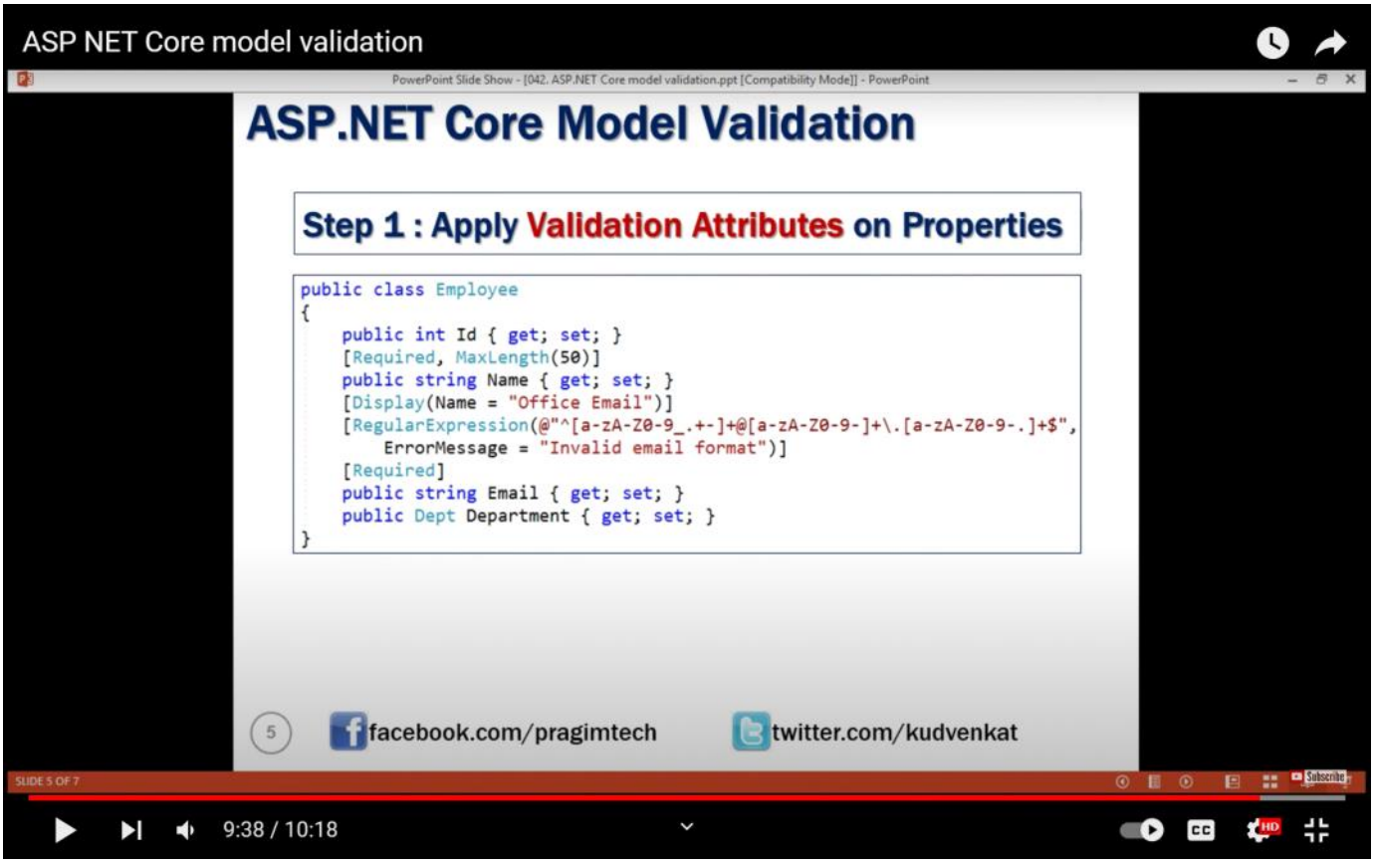


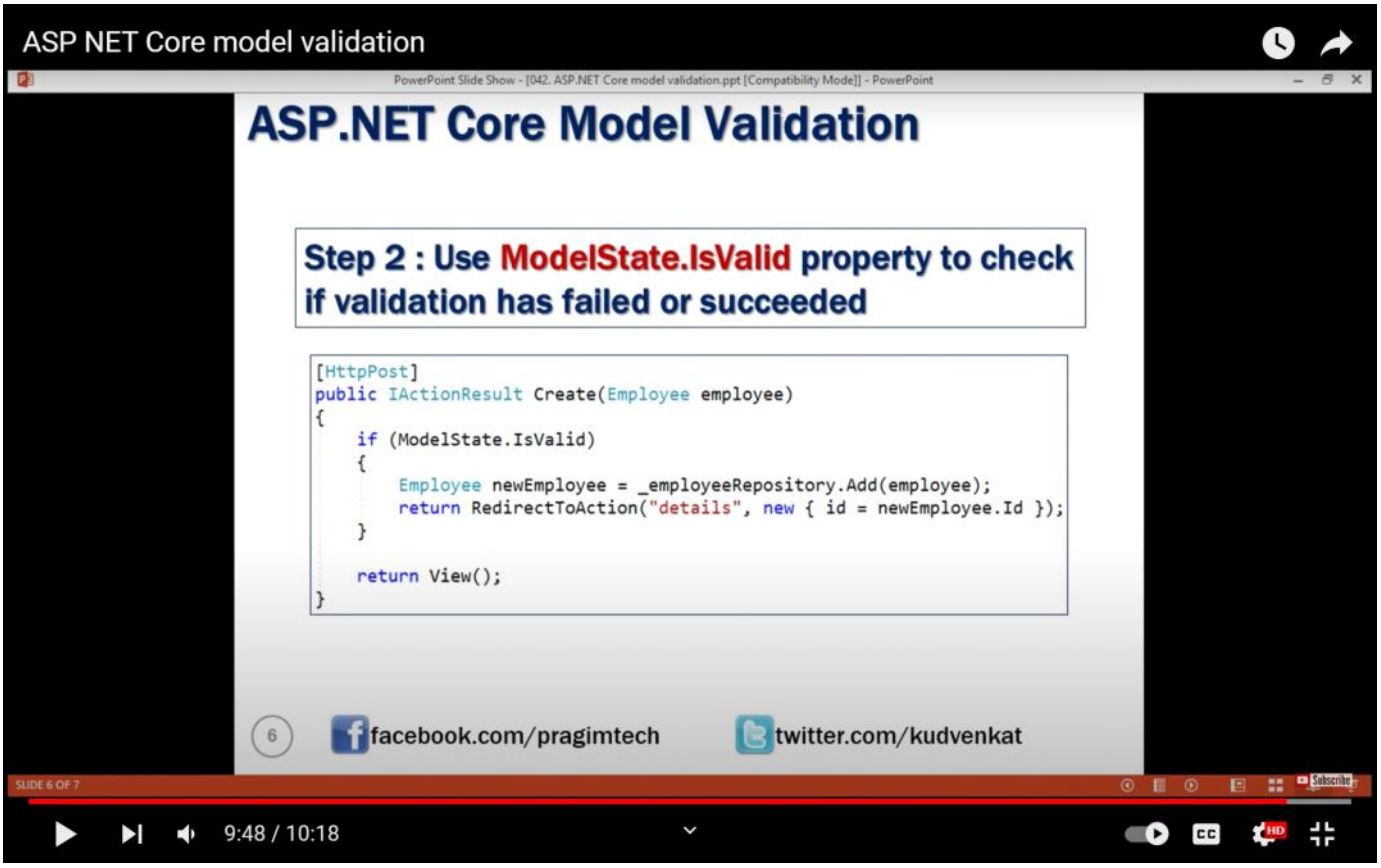
Enabling and understanding client-side validation in ASP.NET Core MVC.
Using jQuery Validation and Unobtrusive JavaScript for seamless client-side validation.
Customizing client-side validation behaviors.

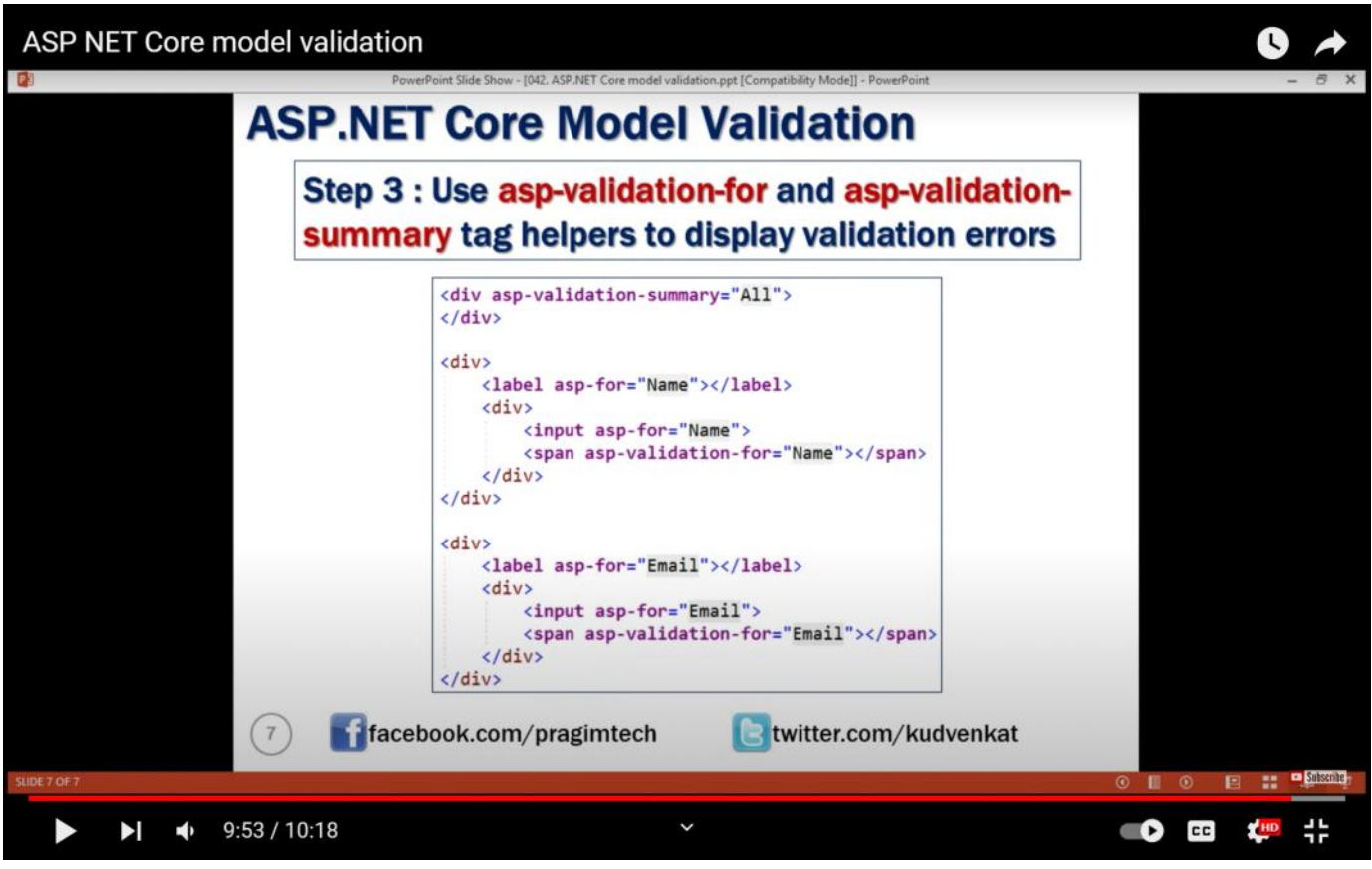
Validations

Thursday, February 1, 2024 2:11 PM






ASP NET Core model validation
PowerPoint Slide Show - [042. ASP.NET Core model validation.ppt [Compatibility Mode]] - PowerPoint
ASP.NET Core Model Validation
Step 2 : Use ModelState.IsValid property to check if validation has failed or succeeded
[HttpPost]
public IActionResult Create(Employee employee)
{
if (ModelState.IsValid)
{
Employee newEmployee = _employeeRepository.Add(employee);
return RedirectToAction("details", new { id = newEmployee.Id });
}
return View();
}
6
facebook.com/pragimtech
twitter.com/kudvenkat
SLIDE 6 OF 7
9:48 / 10:18


ASP NET Core model validation
PowerPoint Slide Show - [042. ASP.NET Core model validation.ppt [Compatibility Mode]] - PowerPoint
ASP.NET Core Model Validation
Step 3 : Use asp-validation-for and asp-validation-summary tag helpers to display validation errors

<label asp-for="Name"></label>

<input asp-for="Name">

<label asp-for="Email"></label>

<input asp-for="Email">

7
facebook.com/pragimtech
twitter.com/kudvenkat
SLIDE 7 OF 7
9:53 / 10:18

Mobile Number Validations :

```
 using System.ComponentModel.DataAnnotations;

public class UserModel
{
   [Required]
   [Display(Name = "Mobile Number")]
   [RegularExpression(@"^[789]\d{9}$", ErrorMessage = "Invalid mobile number")]
   public string MobileNumber { get; set; }
}
```

Creadit Card Form Validation :

```
public class PaymentModel
{
   [Required]
   [RegularExpression(@"^\d{16}$", ErrorMessage = "Credit card number must be 16 digits")]
   public string CreditCardNumber { get; set; }
}
```

Email Validations :

```
public class EmailConfirmationModel
{
   [Required]
   [EmailAddress]
   [Display(Name = "Email")]
   public string Email { get; set; }

   [Required]
   [EmailAddress]
   [Display(Name = "Confirm Email")]
   [Compare("Email", ErrorMessage = "The email and confirmation email do not match.")]
   public string ConfirmEmail { get; set; }
}
```

Custom Error Messages and Display Names

```
public class RegistrationModel
{
   [Required(ErrorMessage = "Please enter your first name")]
   [Display(Name = "First Name")]
   public string FirstName { get; set; }

   [Required(ErrorMessage = "Please enter your last name")]
   [Display(Name = "Last Name")]
   public string LastName { get; set; }

   [Required]
   [EmailAddress(ErrorMessage = "Invalid Email Address")]
   public string Email { get; set; }
}
```

Range Validations :

```
public class ProductModel
{
   [Required]
   public string Name { get; set; }

   [Range(1, 1000, ErrorMessage = "Quantity must be between 1 and 1000")]
   public int Quantity { get; set; }

   [Range(0.01, 10000.00, ErrorMessage = "Price must be between $0.01 and $10,000")]
   public decimal Price { get; set; }
}
```

Configuration Sources

Wednesday, January 31, 2024 1:51 PM

Configuration Sources

- appsettings.json: Understand how to use and format this JSON file for storing configuration settings.
- Environment-Specific Settings: Learn about appsettings.{Environment}.json files (like appsettings.Development.json, appsettings.Production.json) for environment-specific configurations.
- User Secrets: This is used in development to store sensitive data outside of your project.
- Environment Variables: Learn how these can be used to override settings in other configuration sources.
- Command-Line Arguments: Understand how to read configuration values passed from the command line.
- Custom Providers: Study how to create and use custom configuration providers.

APP-SETTINGS.JSON :

Imagine you have a toy robot, and you can tell it to do different things by writing down instructions on little cards and putting them into a special box. Whenever you want the robot to do something, it looks in the box, reads your instructions, and knows what to do.
The appsettings.json file is just like that box. It's a special file where developers write down settings or instructions for their computer program. These settings can be things like:

- **What color you want the background of your game to be.**
- **How loud the music should play.**
- **Where to find the list of all the levels in your game.**

Here's how it works:

1. **It's a Text File: Just like writing on a piece of paper, appsettings.json is a text file where you can write things down.**
2. **It Uses JSON Format: JSON is a way of writing down information so that both humans and computers can understand it easily. It's like using simple**

symbols and words to tell your toy robot what to do. In JSON, you write instructions in pairs, like "background: blue" to tell the program, "please make the background blue."

3. **It's Organized in Sections: In this magic box, you can have different sections, like one for game settings, one for sound levels, and so on. It's like having different compartments in the box for different types of instructions.**
4. **The Program Reads It: When your computer program starts, it looks inside appsettings.json to see what settings or instructions you've written. Then, it uses these settings to decide how to run or what to show.**
5. **You Can Change It: If you want to change something about your program, like the background color or the music volume, you just change the instructions in the appsettings.json file, and the next time the program runs, it will follow these new instructions.**

Create Different appsettings Files :

Imagine you have a robot that helps you with different tasks. This robot can work in different places, like your home, school, or a park. Depending on where the robot is, it needs different instructions. For example, at home, it might need to know where your toys are kept, but at school, it needs to know where the books are.

In computer programs, we sometimes need to give the program different instructions depending on where it's being used. For example, when a program is being tested by developers, it might need one set of instructions, and when it's being used by real users, it needs another set.

This is where appsettings.{Environment}.json files come in. They are like different sets of instructions for your program, depending on where it's being used. The {Environment} part in the file name changes based on the environment. Common environments are:

- **Development: This is like the robot being at home. It's where programmers are still working on the program, testing and fixing it. The settings here might be for testing purposes, like where to find test data. The file name would be appsettings.Development.json.**
- **Production: This is like the robot working at a park where everyone can see it. It's the real setting where real users use the program. The settings in appsettings.Production.json are for this real-world use, like connecting to the actual database where all the important data is kept.**
- **Staging, Testing, QA (Quality Assurance): These are like other places where the robot might work, each with its own set of instructions. These environments are used for specific types of testing and preparing the program before it goes to the Production environment.**

- **appsettings.json: This is the default configuration file.**
- **appsettings.Development.json: This file contains configuration settings specific to the development environment.**
- **appsettings.Production.json: This file contains settings for the production environment.**

the AddDbContext method uses the DefaultConnection connection string. Depending on the environment, it will automatically pick the connection string from either appsettings.Development.json or appsettings.Production.json.

USER SECRETS :

Alright, let's explain "User Secrets" in ASP.NET Core like I would to a small child:
Imagine you have a secret treasure map that shows where you've hidden your favorite candies. You don't want your friends to find your candies, so you keep the map hidden and only you know where it is. In the world of computer programming, sometimes we also have important secrets that we don't want to share with others, like passwords or special codes.
When people make computer programs, especially when they are just starting to build them (which we call the "development" phase), they often need to use these special secrets. For example, a program might need a password to talk to a database where it stores all its information.
But there's a problem. If you put these secrets directly in your program's code, then anyone who can see the code can see your secrets. That's like leaving your treasure map out where everyone can find it. This is not safe, especially when you share your code with others or store it somewhere like GitHub, where lots of people can see it.
Here's where "User Secrets" come in. In ASP.NET Core, "User Secrets" is like a special, hidden place on your computer where you can keep these secrets while you're building your program. Only your program knows where this special place is and how to read the secrets from it. This way, you can use all the secrets you need without putting them in your code and without letting other people find them.
So, "User Secrets" helps keep your program's important secrets safe, just like hiding your treasure map in a secret place where only you can find it. This is very important for keeping your program and its information safe and secure.

POCO :

a POCO (Plain Old CLR Object) is primarily considered a model. It's a simple object that doesn't have any special dependencies on frameworks or libraries. It's used to hold data and is often used in applications to represent the structure of your data without adding additional complexity.

Step 1: Define the Configuration Model

```
public class AppSettings
```

```
{
   public string Title { get; set; }
   public bool UpdatesEnabled { get; set; }
   public int MaxItems { get; set; }
}
```

Step 2: Bind Configuration in Program.cs

```
        var builder = WebApplication.CreateBuilder(args);

        // Bind and register AppSettings
        var appSettings =
        builder.Configuration.GetSection("AppSettings").Get<AppSettings>();
        builder.Services.AddSingleton(appSettings);

        var app = builder.Build();

        // Configure the HTTP request pipeline and endpoints
        if (!app.Environment.IsDevelopment())
        {
           app.UseExceptionHandler("/Home/Error");
        }

        app.UseStaticFiles();

        app.UseRouting();

        app.UseAuthorization();

        app.MapControllerRoute(
           name: "default",
           pattern: "{controller=Home}/{action=Index}/{id?}");

        app.Run();
```

Step 3: Access Configuration in a Controller

```
using Microsoft.AspNetCore.Mvc;

public class HomeController : Controller
{
   private readonly AppSettings _appSettings;

   public HomeController(AppSettings appSettings)
   {
      _appSettings = appSettings;
   }

   public IActionResult Index()
   {
      // Pass configuration data to the view
      ViewBag.Title = _appSettings.Title;
      ViewBag.UpdatesEnabled = _appSettings.UpdatesEnabled;
      ViewBag.MaxItems = _appSettings.MaxItems;
```

```
        return View();
    }
}
```

Step 4: Create a View to Display Configuration Data

```
@{
    ViewData["Title"] = "Home Page";
}

<h1>@ViewBag.Title</h1>
<p>Updates Enabled: @ViewBag.UpdatesEnabled</p>
<p>Max Items: @ViewBag.MaxItems</p>
```

APPSETTINGS.JSON

```
{
  "AppSettings": {
    "Title": "My MVC Application",
    "UpdatesEnabled": true,
    "MaxItems": 5
  }
}
```

DYNAMIC Cofigurations :

, let's simplify this and imagine you have a magical book that tells stories. Every time you open the book, it can tell the story in a slightly different way depending on what you're feeling like that day. Now, in the world of computer programs, especially in websites made with ASP.NET Core, we sometimes need our program to be able to change how it does things while it's still running, based on new information or instructions it gets. This is a bit like the magical book changing its story.

IOptionsSnapshot :

IOptionsSnapshot is used to read settings that can change during the application's lifetime. It's great for web requests because it gets the latest settings at the start of each request. It's like checking the magical book each time you open it to see if the story has changed.

Step 1: Create the Configuration Class

```
public class GreetingSettings
{
    public string Message { get; set; } = "Hello, World!";
}
```

Update the AppSettings.Json :

```
{
  "GreetingSettings": {
    "Message": "Hello, everyone!"
  }
}
```

Step 3: Modify the Startup.cs or Program.cs

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.Configure<GreetingSettings>
(builder.Configuration.GetSection("GreetingSettings"));
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
   app.UseExceptionHandler("/Home/Error");
   app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
   name: "default",
   pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

STeP:4 :

INDEX.CSHTML :

```
@{
   ViewData["Title"] = "Home Page";
}

<div class="text-center">
   <h1 class="display-4">Welcome</h1>
   <p>@ViewBag.Message</p>
</div>
```

Step 4: Create the HomeController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Options;

public class HomeController : Controller
{
   private readonly IOptionsSnapshot<GreetingSettings> _greetingOptions;

   public HomeController(IOptionsSnapshot<GreetingSettings> greetingOptions)
```

```
    {
        _greetingOptions = greetingOptions;
    }

    public IActionResult Index()
    {
        var message = _greetingOptions.Value.Message; // Access the current value of the Message
        return Content(message);
    }
}
```

Step 5: Run Your Application

When you run your application and navigate to the home page, it should display "Hello, everyone!" from the appsettings.json.

Dynamically Updating the Message
If you change the Message value in appsettings.json while the application is running, you will need to restart the application to see the change if you're using IOptionsSnapshot. For changes without restarting, consider more advanced scenarios involving IOptionsMonitor

IOptionsMonitor

IOptionsMonitor is a bit more powerful. It not only gets the latest settings like IOptionsSnapshot, but it can also notify our program whenever the settings change, without needing to start a new request. It's like the magical book whispering to us that it has a new story to tell, even when we're still reading it.

Step 1: Create the Configuration Class

```
public class MyConfig
{
    public string Message { get; set; } = "Default Message";
}
```

Step 2: Add Configuration in appsettings.json

```
{
  "MyConfig": {
    "Message": "Hello, World!"
  }
}
```

Step 3: Configure IOptionsMonitor in Program.cs (for .NET 6 and later)

```
var builder = WebApplication.CreateBuilder(args);

// Bind MyConfig to the configuration section
builder.Services.Configure<MyConfig>(builder.Configuration.GetSection("MyConfig"));

// Add services to the container.
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

Step 4: Create the HomeController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Options;

public class HomeController : Controller
{
    private readonly IOptionsMonitor<MyConfig> _configMonitor;

    public HomeController(IOptionsMonitor<MyConfig> configMonitor)
    {
        _configMonitor = configMonitor;
    }
```

```
    public IActionResult Index()
    {
        var message = _configMonitor.CurrentValue.Message; // Access the current value of the
Message
        return View("Index", message);
    }
}
```

Step 5: Create the View

```
@model string

@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>@Model</p>
</div>
```

Dynamically Updating the Message

To see IOptionsMonitor in action, change the value of Message in appsettings.json while the application is running. In development environments, the app will pick up the changes without needing to restart, and the new message will be displayed when you refresh the page.

SECRET KEY IN THE MVC :

Step 1: Enable User Secrets in Your Project

```
<PropertyGroup>
  <UserSecretsId>your-unique-id</UserSecretsId>
</PropertyGroup>
```

Step 2: Add a Secret Key or Set Secret key into that page of secret key

```
dotnet user-secrets set "SecretKey" "your-secret-value"
```

Step 3: Accessing the Secret Key in Your Application

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;

public class ExampleController : Controller
{
```

```
    private readonly IConfiguration _configuration;

    public ExampleController(IConfiguration configuration)
    {
        _configuration = configuration;
    }

    public IActionResult Index()
    {
        // Retrieve the secret key
        string secretKeyValue = _configuration["SecretKey"];

        // Use the secret key as needed (e.g., as part of your application's logic)

        return View();
    }
}
```

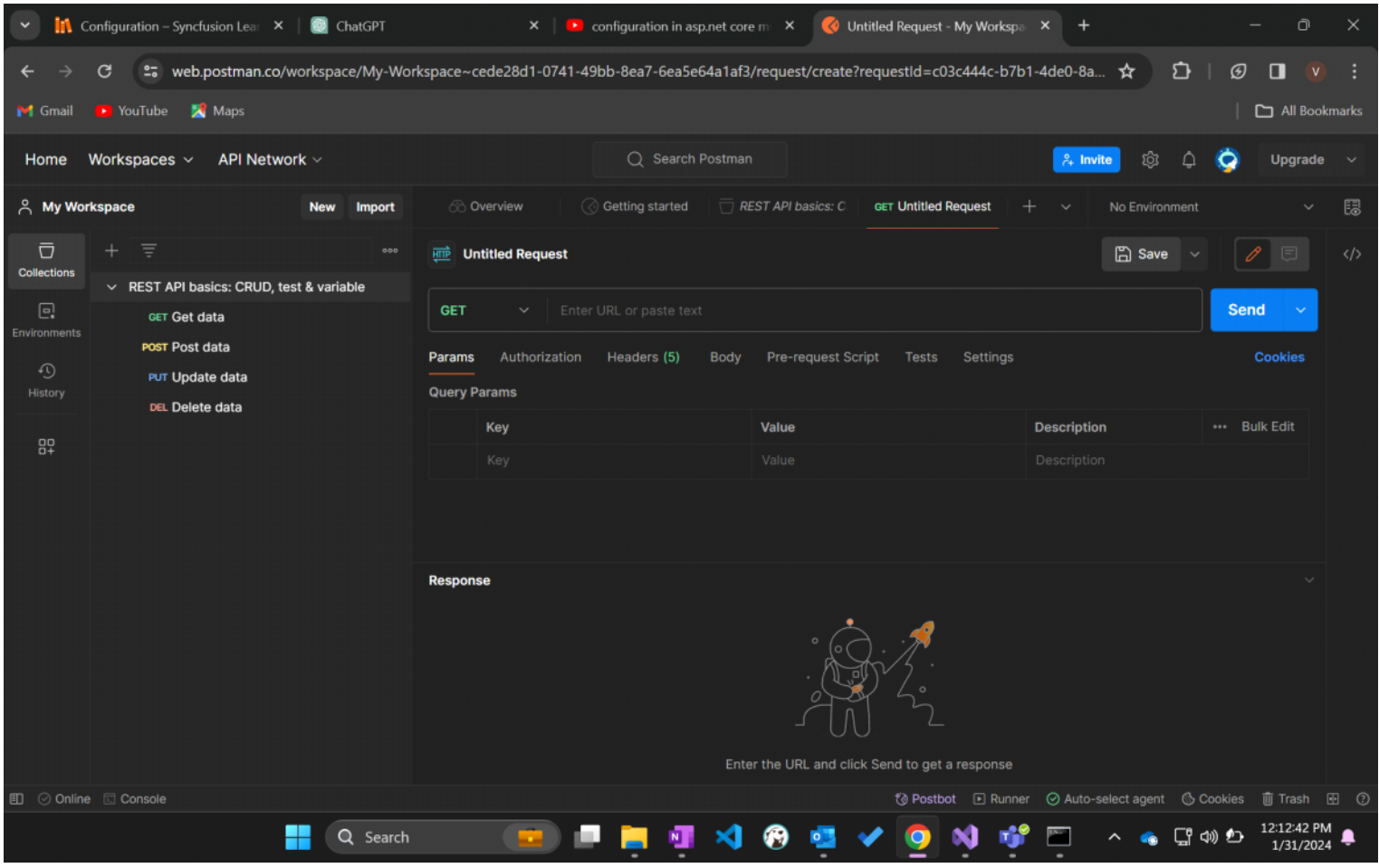
Weather Apl

Wednesday, January 31, 2024 12:13 PM

Weather Api contains all the details of weathers by based on the locations in the form of weather api .

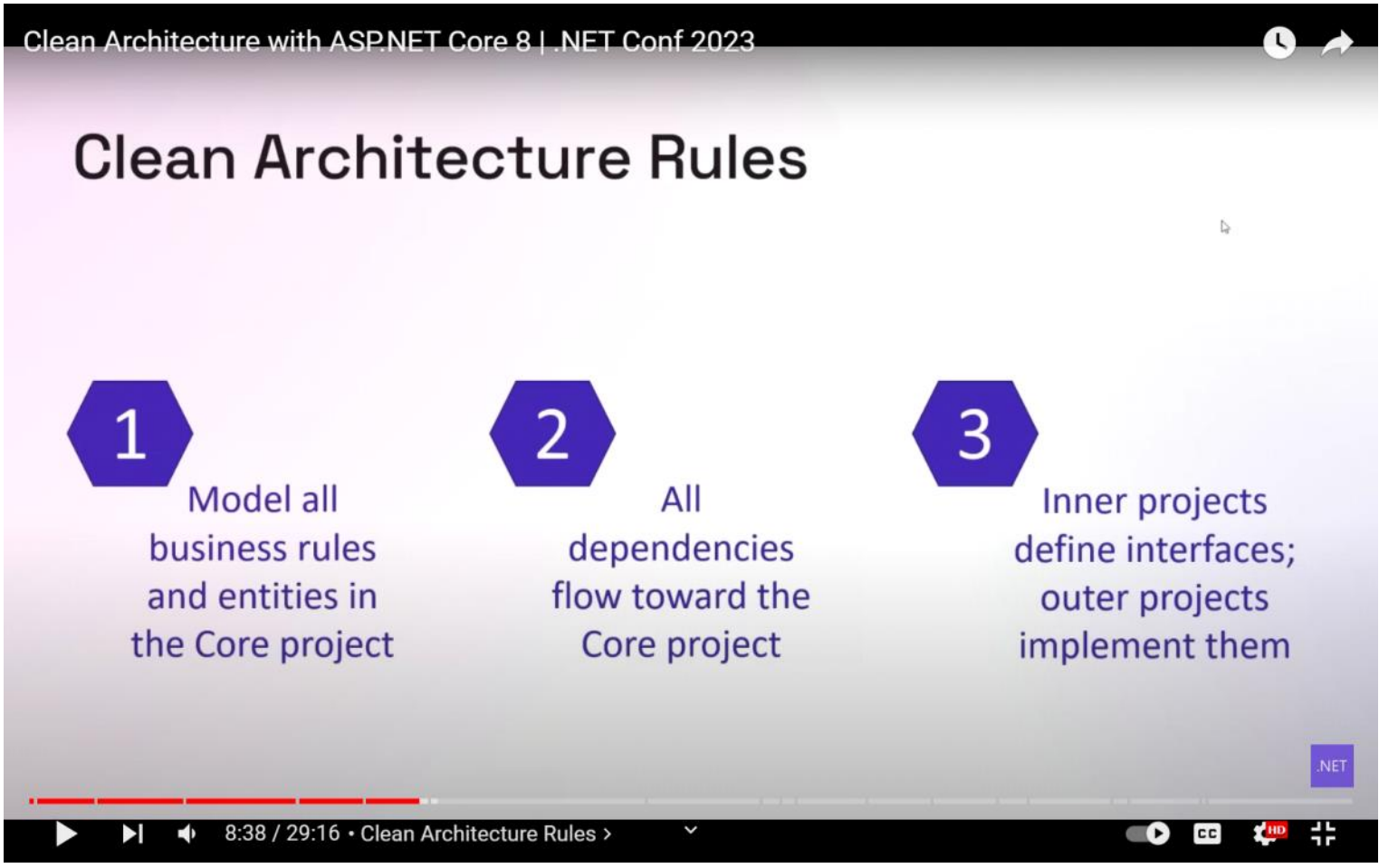
POST MAN API

Wednesday, January 31, 2024 12:12 PM



Clean Archietecture

Wednesday, January 31, 2024 12:10 PM



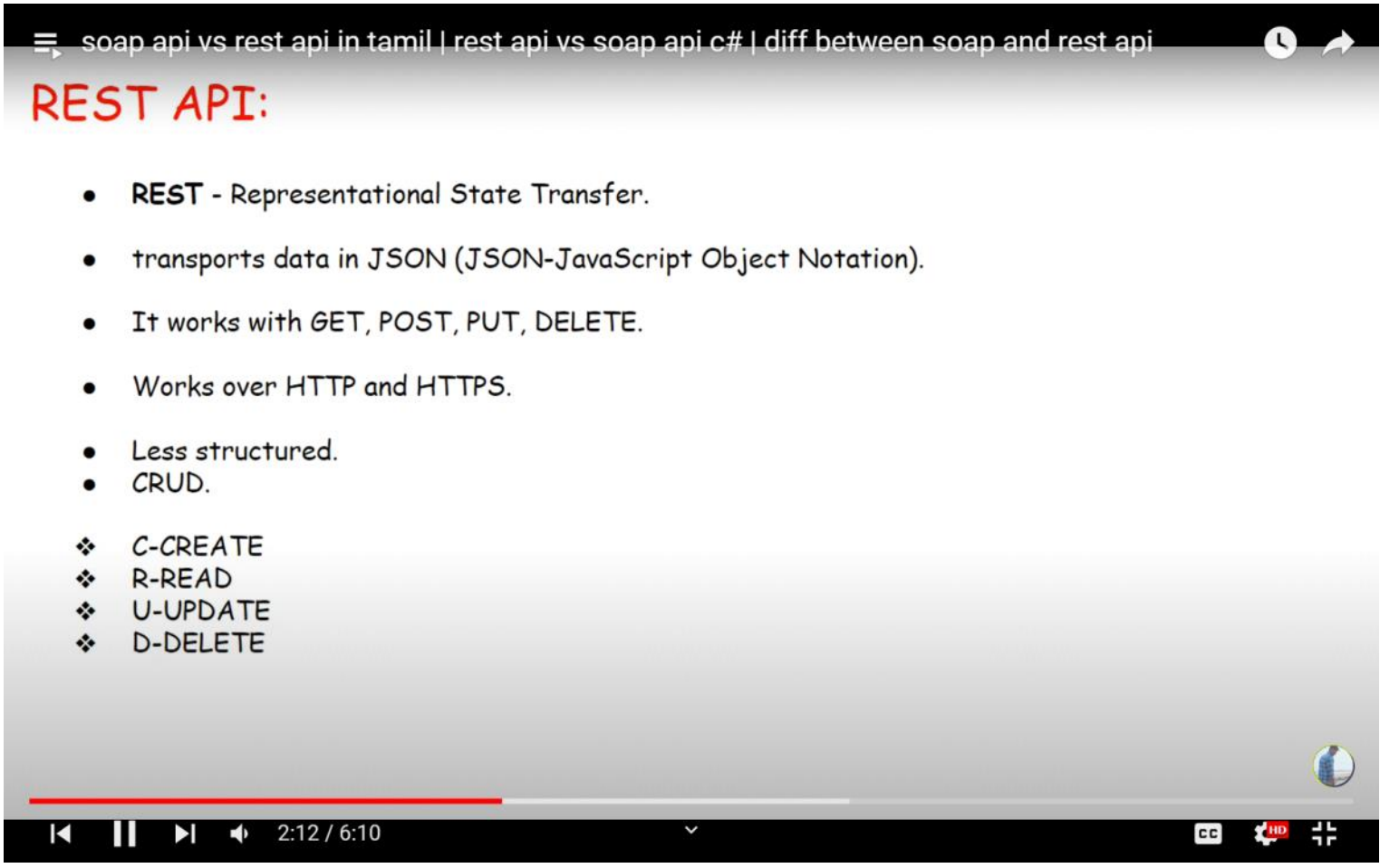
Soap API vs Web API

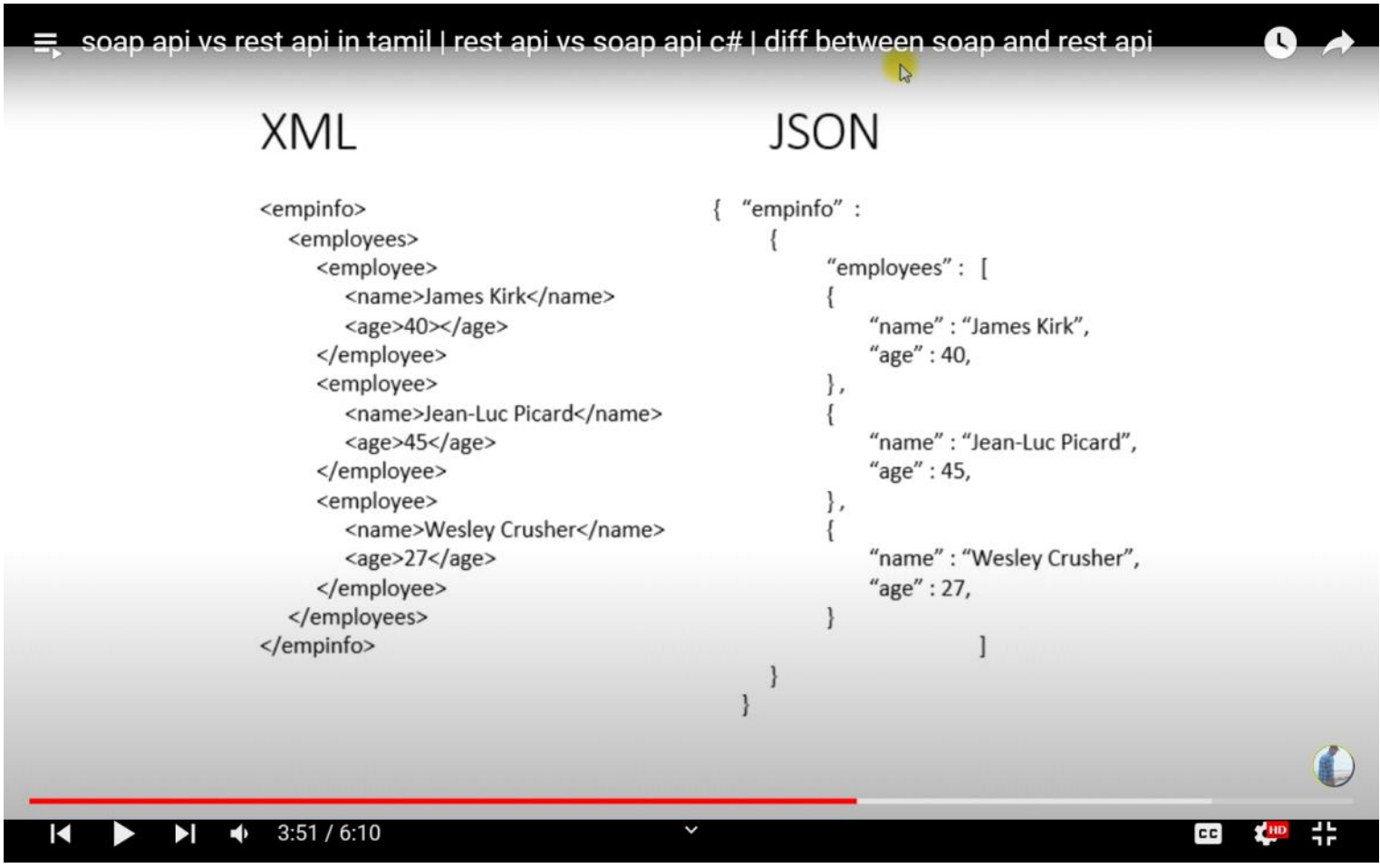
Wednesday, January 31, 2024 12:00 PM

Soap Api :

SOAP API:

- **SOAP** - Simple Object Access Protocol.
- Transports data in standard XML format.(XML-Extensible Markup Language).
- Because it is XML based and relies on SOAP, it works with WSDL(WSDL-Web Services Description Language).
- Works over HTTP, HTTPS, SMTP, XMPP.
- Highly structured.


soap api vs rest api in tamil | rest api vs soap api c# | diff between soap and rest api
REST API:
REST - Representational State Transfer.
transports data in JSON (JSON-JavaScript Object Notation).
It works with GET, POST, PUT, DELETE.
Works over HTTP and HTTPS.
Less structured.
CRUD.
C-CREATE
R-READ
U-UPDATE
D-DELETE
2:12 / 6:10


soap api vs rest api in tamil | rest api vs soap api c# | diff between soap and rest api
XML
JSON
<empinfo>
<employees>
<employee>
<name>James Kirk</name>
<age>40</age>
</employee>
<employee>
<name>Jean-Luc Picard</name>
<age>45</age>
</employee>
<employee>
<name>Wesley Crusher</name>
<age>27</age>
</employee>
</employees>
</empinfo>
{ "empinfo" :
{
"employees" : [
{
"name" : "James Kirk",
"age" : 40,
},
{
"name" : "Jean-Luc Picard",
"age" : 45,
},
{
"name" : "Wesley Crusher",
"age" : 27,
}
]
}
}
3:51 / 6:10

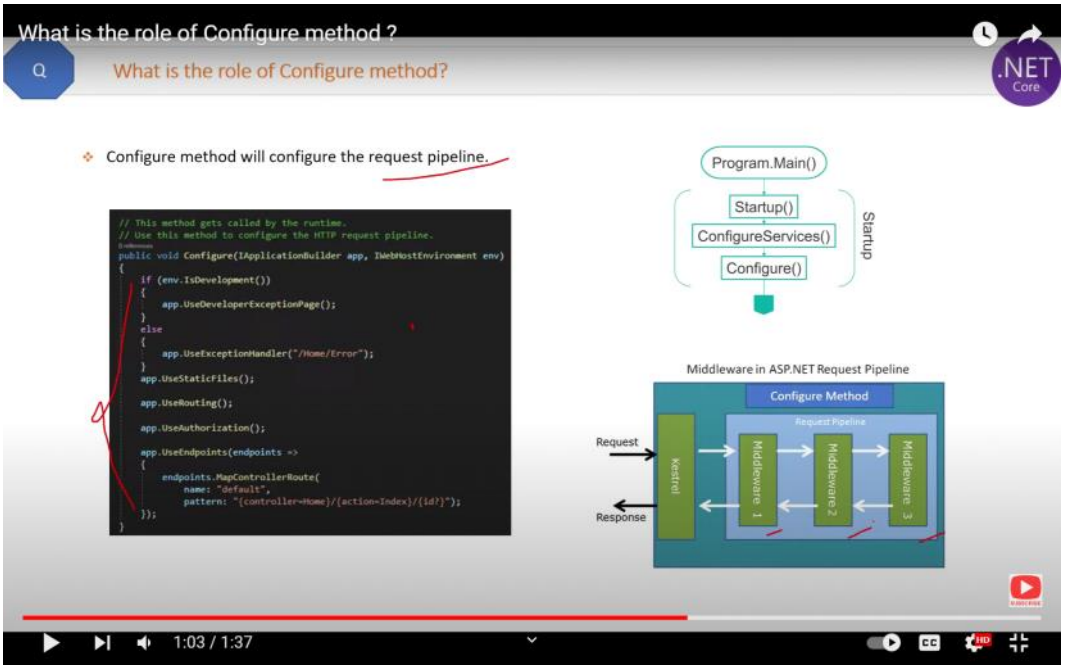
Configuration

Wednesday, January 31, 2024 11:42 AM

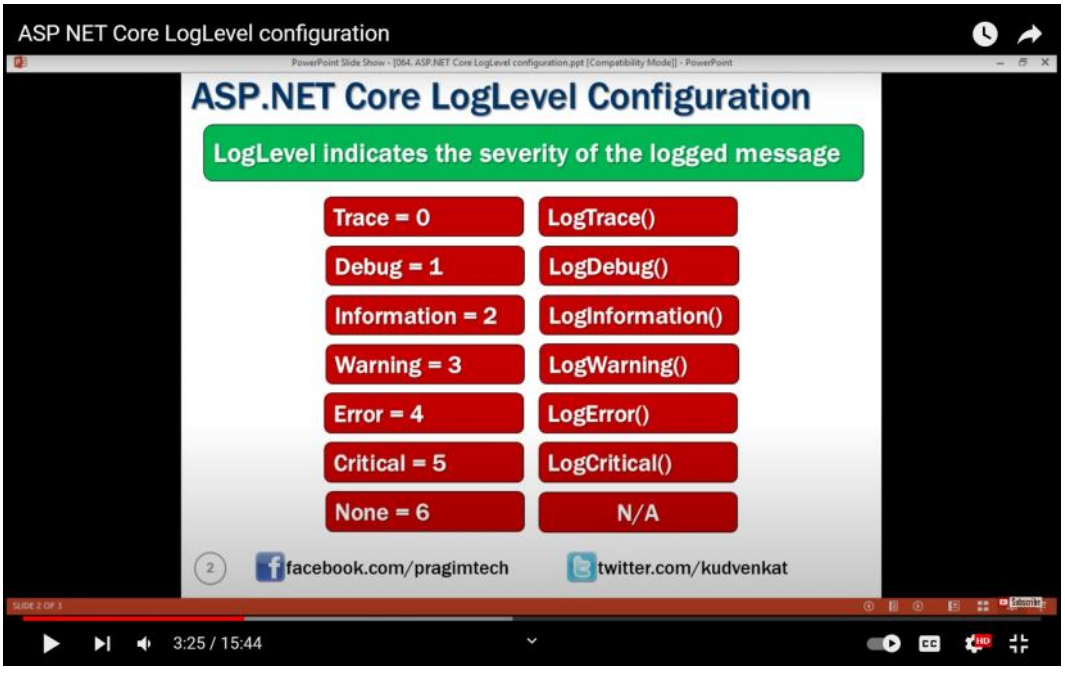
Default application configuration sources

```
var builder = WebApplication.CreateBuilder(args);
```

Role Of Configure Method :

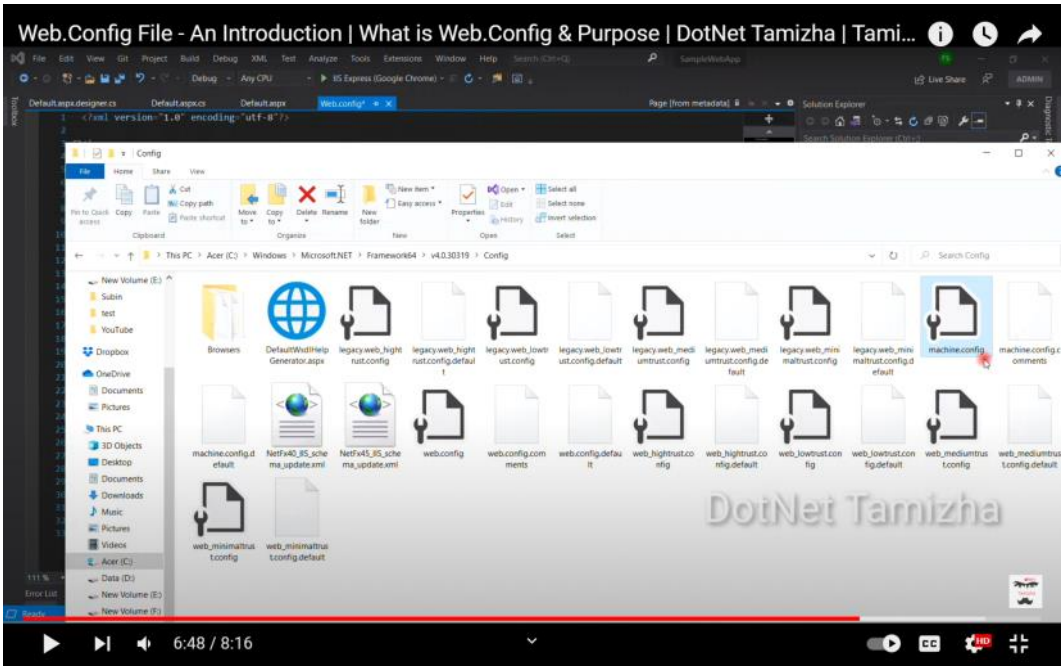


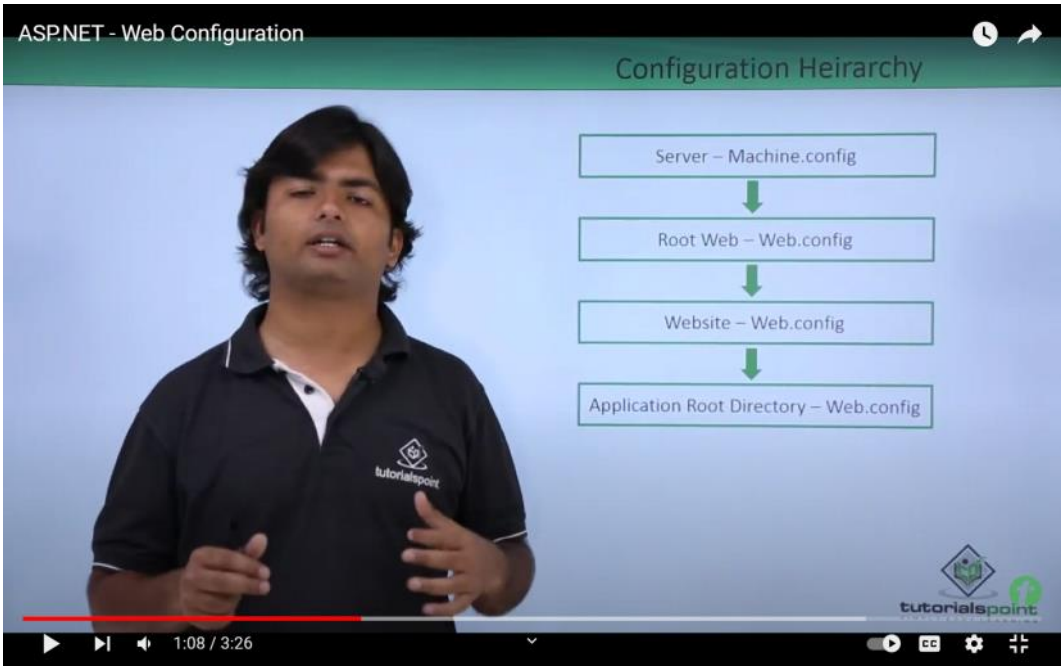
Log Level Configuration :



WEB.CONFIG :

If web.config is not available or deleted .Machine.config will be available





FallBack

Wednesday, January 31, 2024 9:45 AM

Think about a fallback value like having a backup plan. For example, imagine you want to go to the park, but if it's raining, your backup plan is to stay home and play with your toys.
In the world of computers, a fallback value is like a backup plan for the computer. If the computer is trying to do something but can't do it the way it's supposed to, it uses the fallback value as the backup plan. It's a way to make sure the computer has a good option to use if the first plan doesn't work out. Just like you have a plan to play with toys if you can't go to the park.

Wednesday, January 31, 2024 9:26 AM

CONFIGURATIONS :

Okay! Imagine you have a box of crayons. "Configurations" are like deciding which colors to use and how to use them to make a pretty picture. It's like choosing to color the sky blue and the grass green. In the same way, configurations in things like computers or toys are about choosing how to set them up so they work the way we like.

Model :

```
public class MailSettings
{
   public string MailServer { get; set; }
   public int MailPort { get; set; }
   public string SenderName { get; set; }
   public string SenderEmail { get; set; }
}
```

Build Service in program.cs :

```
builder.Services.Configure<MailSettings>(builder.Configuration.GetSection("MailSettings"));
```

AppSettings :

```
{
  "MailSettings": {
    "MailServer": "smtp.example.com",
    "MailPort": 587,
    "SenderName": "Example Sender",
    "SenderEmail": "sender@example.com"
  }
}
```

Controller :

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Options;
```

```
public class HomeController : Controller
{
   private readonly IOptionsSnapshot<MailSettings> _mailSettings;

   public HomeController(IOptionsSnapshot<MailSettings> mailSettings)
   {
      _mailSettings = mailSettings;
   }

   public IActionResult Index()
   {
      return View(_mailSettings.Value);
   }
}
```

Index.csHtml :

```
@model MailSettings

@{
   ViewData["Title"] = "Mail Settings";
}

<h2>Mail Settings</h2>

<div>
   <p><strong>Mail Server:</strong> @Model.MailServer</p>
   <p><strong>Mail Port:</strong> @Model.MailPort</p>
   <p><strong>Sender Name:</strong> @Model.SenderName</p>
   <p><strong>Sender Email:</strong> @Model.SenderEmail</p>
</div>
```

Logging :

Logging is like keeping a diary or a journal. When you write in your diary about what you did each day, like playing with friends or going to the park, you are keeping a record of your activities.
In the world of computers, logging is similar. Computers keep their own kind of diary. Whenever they do something, like when you play a game or when they check for a virus, they write it down in their log. This helps people understand what the computer has been doing and if there are any problems, just like how your diary helps you remember what you did each day.

App .Use vs App.Run

Tuesday, January 30, 2024 6:38 PM

App. Run ===> only take one parameter

App.use===> many parameters

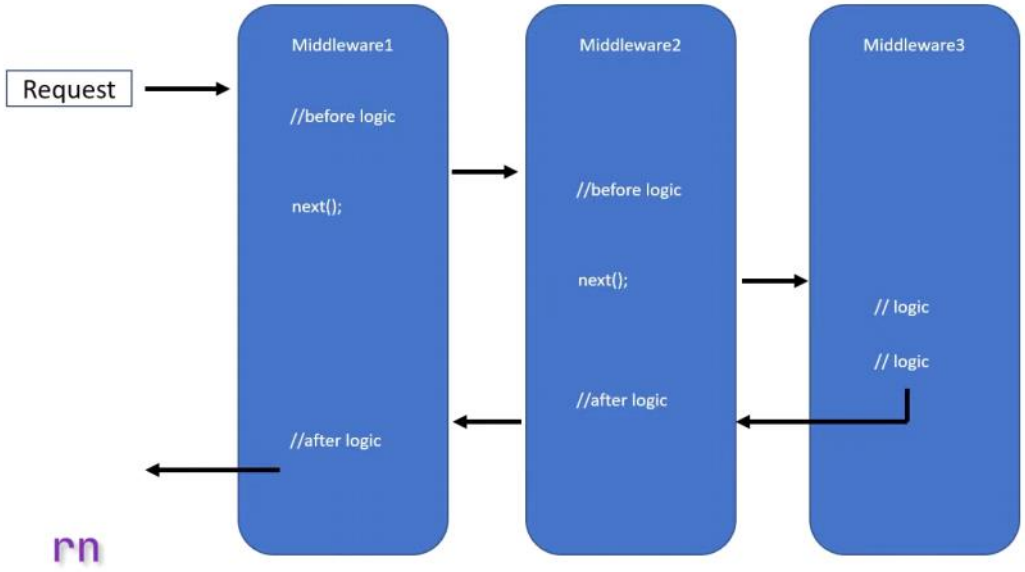
Call Back function

Tuesday, January 30, 2024 5:53 PM

A callback function is like a promise between friends. Imagine you tell your friend, "Hey, after I finish my homework, I'll call you so we can play a game together." Here, you're making a plan to do something (call your friend) after you complete a task (your homework).
In the world of computer programming, a callback function is similar. It's a special function (a set of instructions for the computer) that you ask the computer to run later, after finishing a different task. For example, when you're using a computer or a phone, you might click a button to download a game. The computer will start downloading the game, and you can tell it, "Hey, when you're done downloading, run this callback function to start the game."
So, a callback function is like saying, "Once you're done with this job, here's the next thing I want you to do." It's a way for programmers to tell the computer to wait until one task is finished before starting another one. Just like you and your friend decide to play after finishing your homework!

PipeLine chaining in Middlewares

Tuesday, January 30, 2024 5:18 PM



DateTime.UTC

Tuesday, January 30, 2024 5:16 PM

"DateTime.UtcNow" in programming is like having a special watch that always shows what time it is in a place called Greenwich, England. This place is really important for timekeeping because it's where the whole world decides what time it is everywhere else.
So, when programmers use "DateTime.UtcNow" in their computer code, it's like they're looking at this special watch to find out the exact time right now in Greenwich. This time is called "Universal Coordinated Time" or UTC for short. It's like a standard time that everyone around the world can use to make sure they're all talking about the same moment.
For example, if a programmer in India and a programmer in Canada are working together, they can both use "DateTime.UtcNow" in their code, and they'll know they're talking about the same exact moment, even though it's different local times in India and Canada.
So, "DateTime.UtcNow" is a way for people working with computers to know the exact, standard time no matter where they are in the world. It's like having a universal clock that everyone agrees on!

INVOKE

Tuesday, January 30, 2024 5:14 PM

Alright, let's explain "invoke" in a simple way. Imagine you have a magic wand. When you want something to happen, like cleaning your room or making a toy appear, you wave the wand and say a magic word. By doing this, you're telling the wand what you want it to do, and it starts working its magic to make it happen.
In the world of computers and programming, "invoke" is kind of like using your magic wand. It's a fancy word programmers use when they want to tell the computer to do a specific task or run a special command. Just like how you wave your wand and say a magic word, programmers write a command and tell the computer to run it.
So, when programmers "invoke" something, it's like they're using their magic wand in the computer world to make something happen or to start a specific action.

ILogger

Tuesday, January 30, 2024 5:12 PM

Imagine you're in school and your teacher gives you a special diary. In this diary, you're supposed to write down everything important that happens during the day - like when you learn something new, when you play a game, or even if you have a little problem. At the end of the day, you can read the diary to remember what happened, learn from it, and maybe show it to your parents or teachers.

In the world of computers, especially when people create websites or apps, ILogger is like that special diary. Programmers use ILogger to write down important things that happen when the website or app is running. For example, it can write down when something good happens (like a user successfully logging in), when there's a small problem (like a user entering the wrong password), or even when there's a big problem (like part of the website not working).

Later, programmers can look at what ILogger wrote down to understand how the website or app is working. It helps them figure out if everything is okay, if users are having problems, or if something needs to be fixed.

So, ILogger is a tool that helps programmers keep track of what's happening in their websites or apps, kind of like how you would use a diary to remember and learn from what happens during your day at school.

Delegates

Monday, January 29, 2024 6:42 PM

Real Life Example for a Delegates :

Imagine you have a toy robot that can do different tasks, like picking up toys, playing music, or turning on the lights. But, the robot doesn't decide what to do by itself. You need to tell it which task to do. A delegate is like a remote control for this robot.
When you press a button on the remote, you're telling the robot to do a specific task. Each button on the remote is programmed to tell the robot to do something different. For example, one button might make the robot pick up toys, and another button might make it play music.
In the world of computer programming, a delegate works like this remote control. It's a way for programmers to tell the computer to run a specific piece of code. Just like each button on the remote is connected to a different task for the robot, a delegate in programming is connected to a specific action or function in the code.
So, a delegate is like having a set of instructions that you can tell the computer to run whenever you need. It helps programmers control what the computer does and when it does it, just like how you control your toy robot with its remote control.

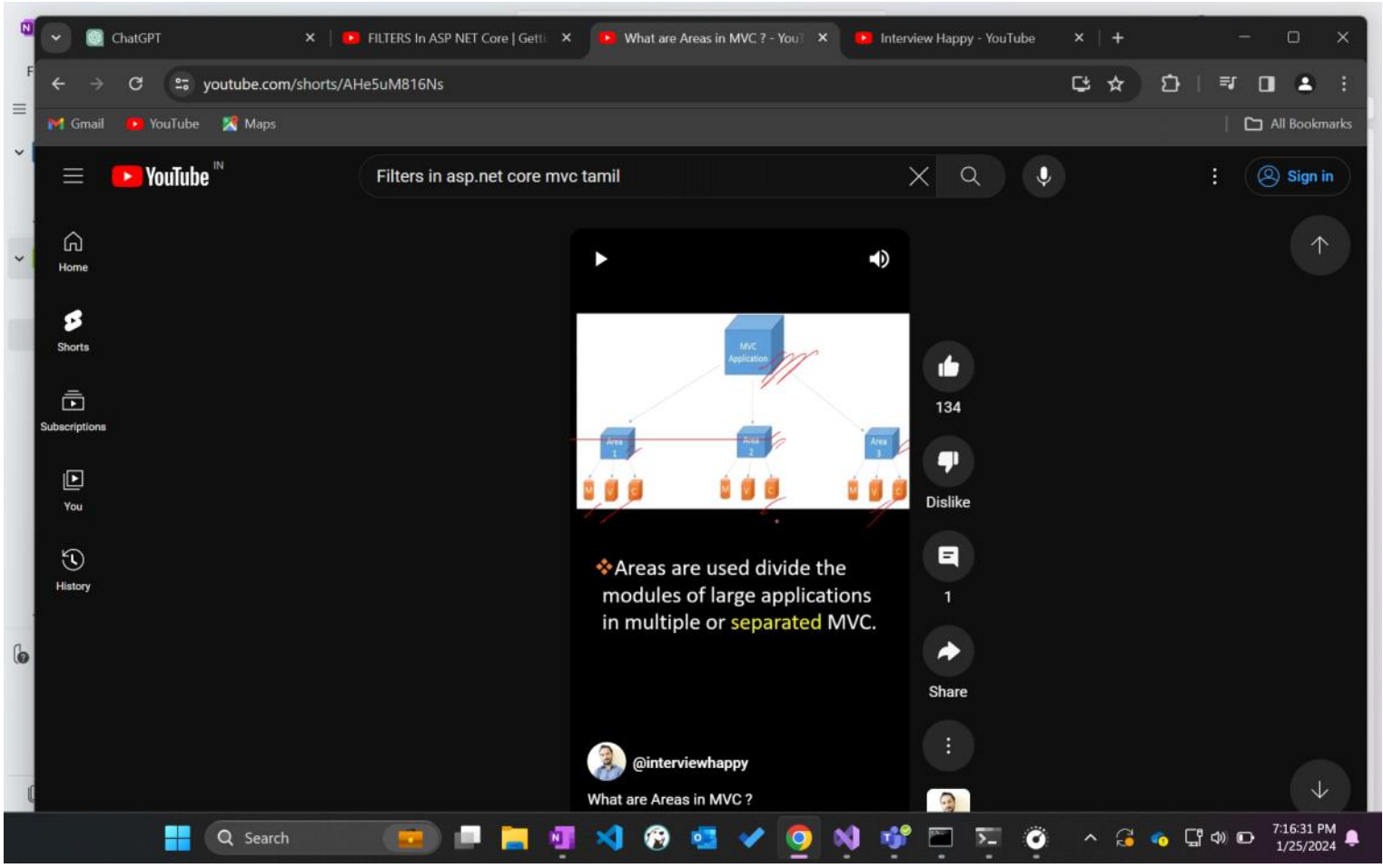
REQUEST DELEGATES :

Request Delegate is a set of instructions that tells each step what to do with the request.

Areas in MVC

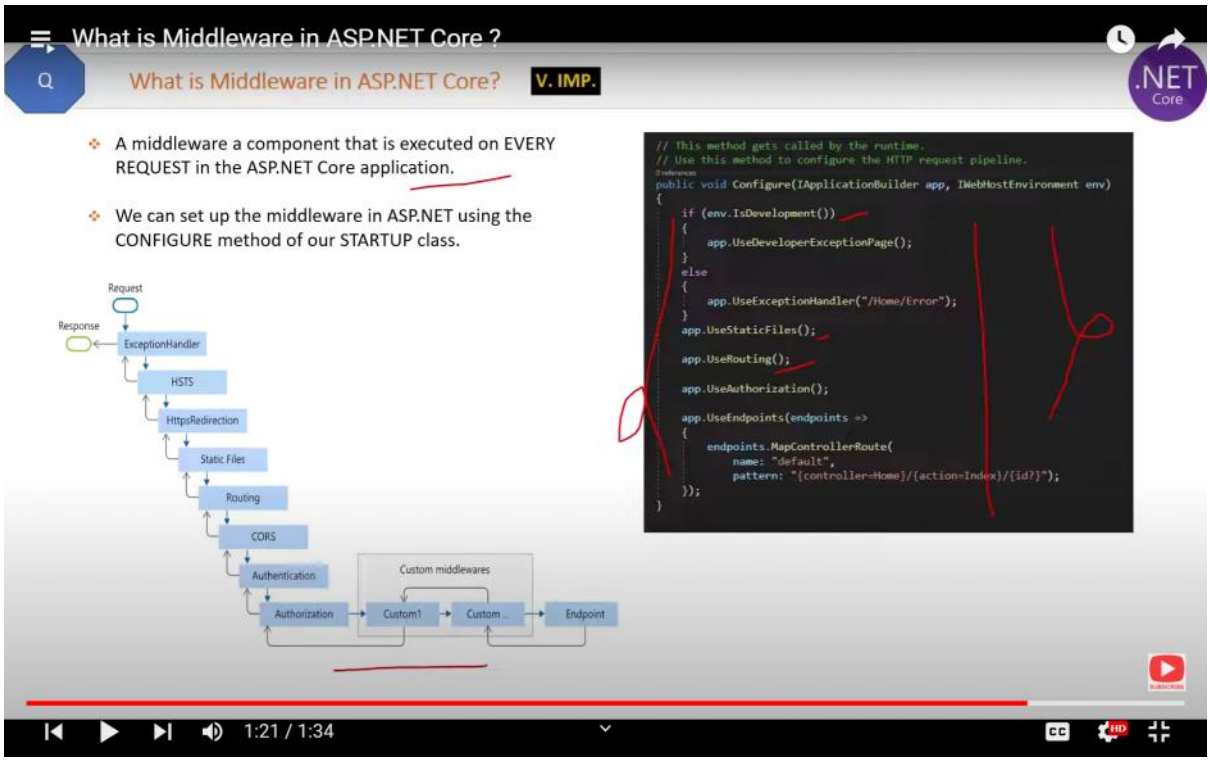
Thursday, January 25, 2024 7:15 PM

Areas are used to separate the big applicatrion into separate modules into smaller parts that is called AREA .Every Area will have containing their own mvc 's

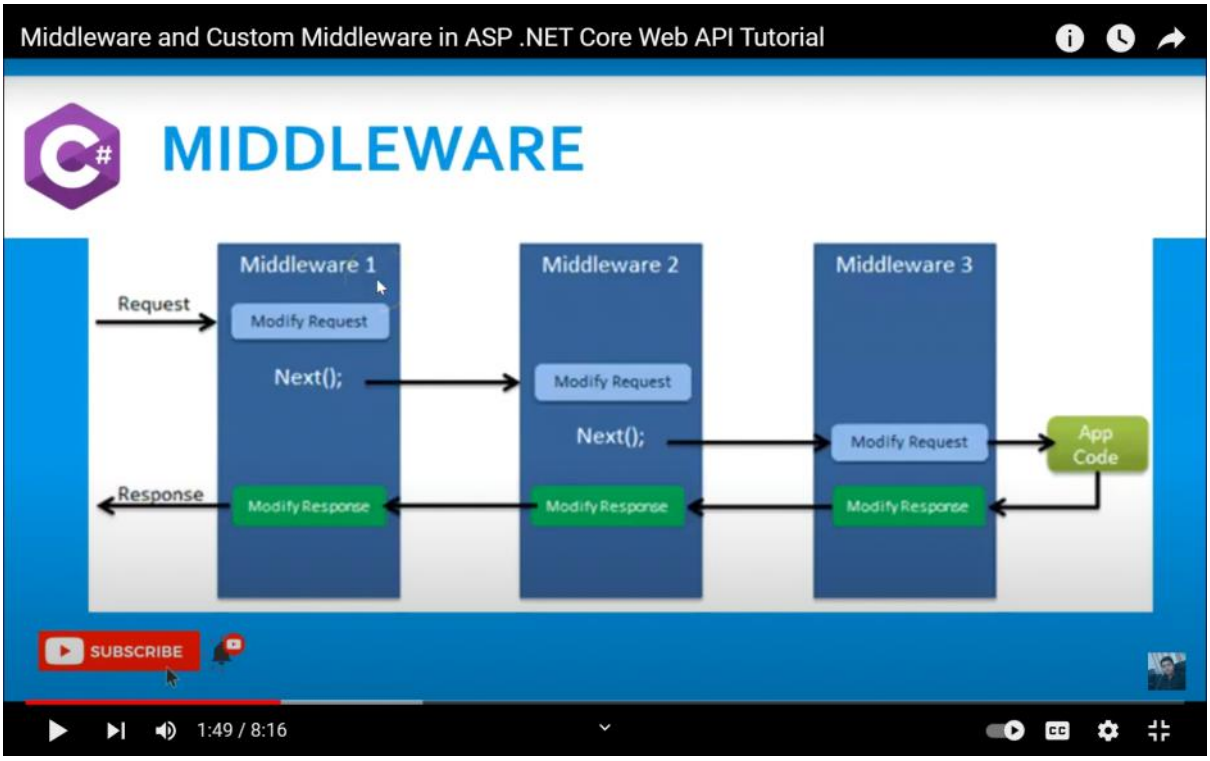


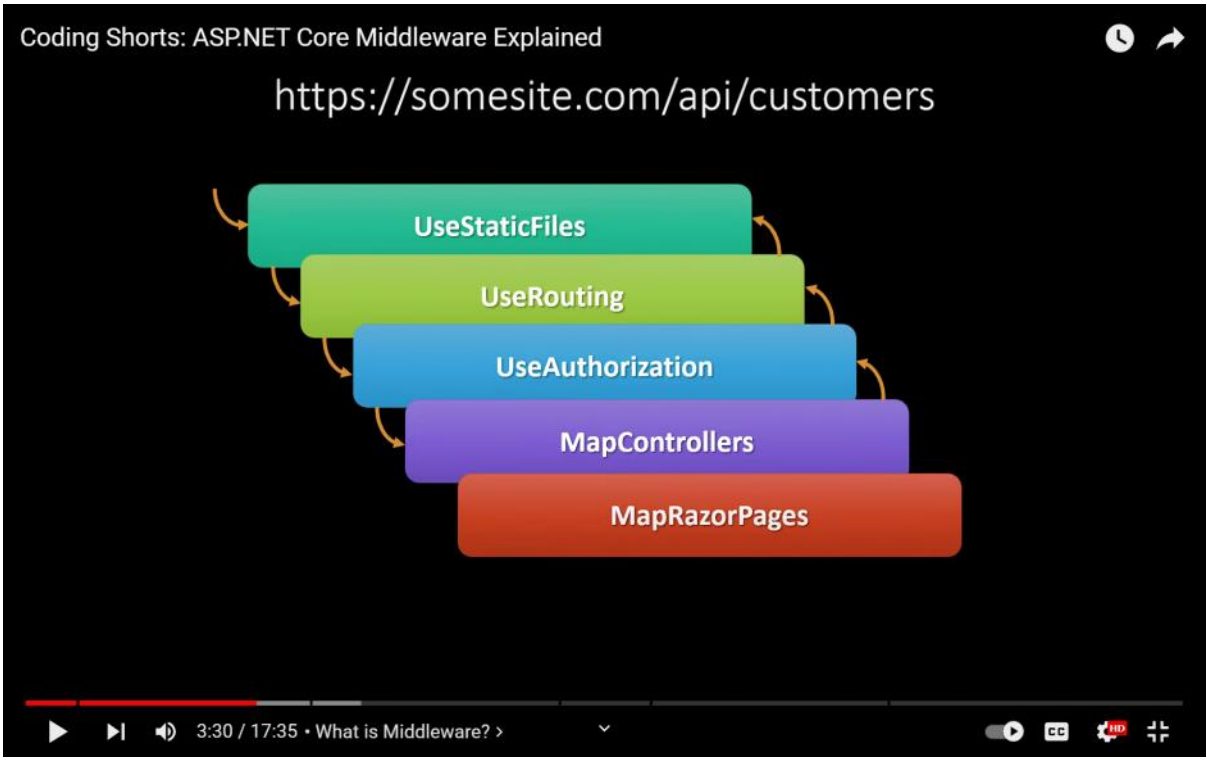
MiddleWares Basic

Thursday, January 25, 2024 7:12 PM



Imagine you're playing with a toy train. The train track is like a website's journey when it gets a request from someone visiting the website. Just like how your train stops at different stations along its track, the website's request goes through different stops before it shows a webpage or information. These stops are what we call "middlewares."


Middleware and Custom Middleware in ASP .NET Core Web API Tutorial
MIDDLEWARE
Middleware 1
Middleware 2
Middleware 3
Request
Modify Request
Next();
Modify Request
Next();
Modify Request
App Code
Response
Modify Response
Modify Response
Modify Response
SUBSCRIBE
1:49 / 8:16


Coding Shorts: ASP.NET Core Middleware Explained
https://somesite.com/api/customers
UseStaticFiles
UseRouting
UseAuthorization
MapControllers
MapRazorPages
3:30 / 17:35 • What is Middleware? ›

Routing In MVC

Thursday, January 25, 2024 6:57 PM

Imagine you're in a big amusement park with lots of different rides, like a roller coaster, a ferris wheel, and a carousel. When you enter the park, you decide which ride you want to go on first. Each ride is in a different part of the park, and there are signs that show you which path to take to get to each ride.
Routing in computers is a lot like finding your way to the different rides in the amusement park. When you use a computer or phone to ask for something on the internet, like opening a website or playing a game, your request is like you deciding which ride you want to go on. Routing is like the signs in the park that show your request the right path to take to get to where it needs to go in the computer.

HTTPSREDIRECTION :

Okay, let's imagine the internet is like a big city with lots of roads. When you want to visit a place in the city, like a toy store or a park, you travel on these roads to get there. Now, some roads are regular roads, and some are special safe roads where there are guards and extra protection to make sure you're safe while you travel.
In the internet world, UseHttpsRedirection is like a helpful guide at the beginning of your journey. When you try to go somewhere on the internet, like a website to watch cartoons or play games, this guide checks the road you're about to use. If you're starting on a regular road, the guide says, "Wait! Let's go on a safe road instead." Then, it takes you to the safe road, which is like using a special protected way to travel on the internet. This safe road is called HTTPS. It's like having a guard with you while you travel, making sure everything is secure and no one can bother you or see where you're going.

Http // 5129 / Home / Index --> Home Controller and that action

```
Public class HomeController : Controller {
        Public viewResult Index(){
                Return View();
        }

}
```

The routing System is responsible for directing incoming requests to the appropriate controller , based on the Url .

This Routing System Typically configured using a routing table which maps URLS to controller Actions .

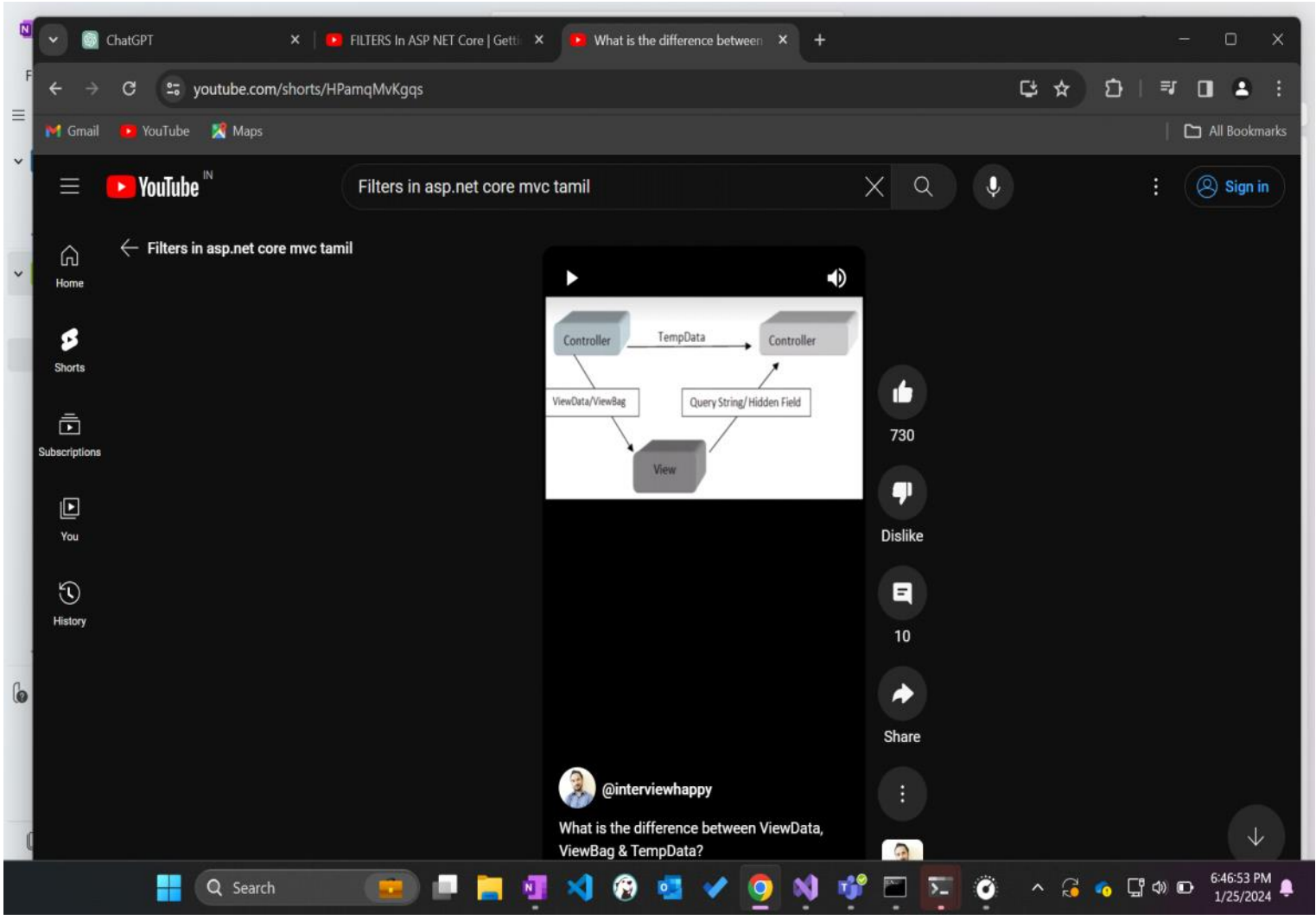
AttriBute Based Routing :

Attribute based routing is used to manipulate the default behaviour of the routing

```
Public class HomeController : Controller {

        [Route("Home/about")]
        Public viewResult GoToAbout(){
                Return View();
        }

}
```

View Data , View Bag , Temp Data

Thursday, January 25, 2024 6:43 PM



View Data and View Bag is used to transfer data from controller to view .

TempData is used to transfer data from controller to controller .

Controller ---> Query, String(Temp Data)---> Controller

Controller ---------->View(Using View Data, View Bag)

Difference Between View Data And View Bag :

Public ActionResult Index(){

```
    ViewBag.Number=42 ;
    ViewData["Number"]=42;
    Return view(); ----> Controller--> View
}
```

.CsHtml

```
<p>@ViewBag.Number</p>
```
---> Here we are not using any typecasting and also it have Type Safe

```
<p>@(int)ViewData["Number"]</p>
```
--->
Here we are explictely write the Type Casting . It doesn't have Type Safe because it is openly type casted .

Different Return Types

Thursday, January 25, 2024 6:33 PM

View	Rendered that view page
Partial View	Rendered within another view . Sometimes one view can have many reusable views .
Json	Returns in the Jsonn format .Mostly used in the WEBAPI's
Content	String xml,Html ,content
File	Return a file content .
RedirectToRoute	Used to go from one action method to another action method
Redirect	Redirect to the url
HTTP NOT FOUND RESULT	Let the browser knows the result source is not found .
Http Status Code Result	Returns the status code
Empty	Returns to be empty

Filters

Thursday, January 25, 2024 5:28 PM

Filters are used to inject logic that runs before or after an action method is executed .

Real Life Example :

Imagine you're making a beautiful drawing. Before you show it to everyone, you want to make sure it's the best it can be. Filters in a website are like special helpers that check and improve your drawing before you show it.

Filters Types :

1.Authorize
2.Resurce
2.Action
3.Result
4.Exception

Global Level Filter :

Without Attribute Keyword It will be Global Filter .
With Attribute Keyword , It will Become a specific Controller , action Filter .
We specify In that Program .cs File .

Add Filter in Program .cs :

```
builder.Services.AddControllersWithViews(options =>
{ options.Filters.Add<GlobalSampleActionFilter>(); });
```

Sequence :

Sequence	Filter scope	Filter method
1	Global	OnActionExecuting
2	Controller	OnActionExecuting
3	Action	OnActionExecuting
4	Action	OnActionExecuted
5	Controller	OnActionExecuted
6	Global	OnActionExecuted

OrderedFilterInterface :

IOrderedfIlterInterface which is used for ordering for ourself .

In order the lower the value it has , the higher the execution it gets .

For Example :
Order - minus kind of things first executed . Then the zero is execute . Here After the + kind of Things will be executed .

Example Code :

```
public class MySampleActionFilter : IActionFilter,IOrderedFilter
{
    private string _name;

    public MySampleActionFilter(string name="Global",int order=0)
    {
        _name = name;
        Order = Order;

    }



    public void OnActionExecuted(ActionExecutedContext context)
    {
        Console.WriteLine($"After {_name}");
    }

    public void OnActionExecuting(ActionExecutingContext context)
    {
        Console.WriteLine($"Before {_name}");
    }

    public int Order { get; set; }
}
```

}

Controller or (Action) Level Filter:

With Attribute Keyword , It will Become a specific Controller , action Filter .
It is specified in that controller

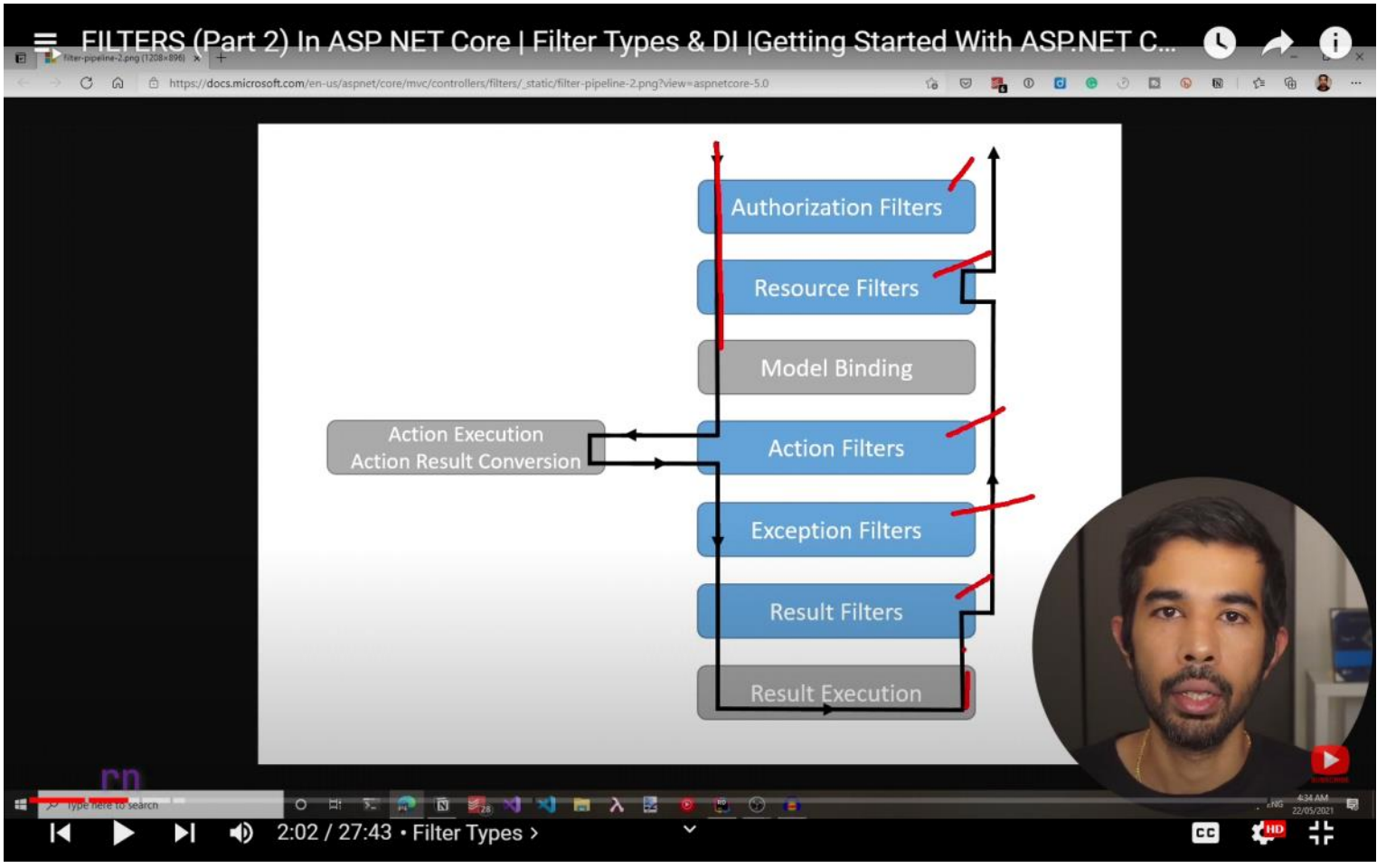
AsyncActionMethod :

AsyncActionFilter :

```
 public async Task  OnActionExecutionAsync(ActionExecutingContext context, ActionExecutionDelegate
next)
 {
    Console.WriteLine($"Before {_name} ");
    //throw new NotImplementedException();
    await next();
    Console.WriteLine($"After {_name}");
 }
```

Filter Types :

Every Filter has Two Methods . But Exception Filter has Only One Method .



3 .Excepton Filter :

ExceptionFilters are used to handle the exceptions that occur during the exception of an action method .
Implementing IExceptionFilter only once ....
It doesn't have before and After events .
Good for trapping exceptions within the actions .

```
[LogExceptionFilter]
Public IActionResult Create(){
 //Loogic for Displaying
}

Public class LogExceptiontFilter :FilterAttribute, IExceptionFilter {
        Public void OnException(ExceptionContext filterContext){
                //This code
```

```
        }
}
```

1 .Authorization Filter :

It Only recommends Before Method not after

Authorization Filters are used to enfore Authentication .
For example if we want to create a create action method .But before that we want to authorize so that

```
[Authorize(Roles="Admin")]
Public IActionResult Create(){

}
```

Here before going into that Create method it must authorize filter roler="admin" and then it goes to that Create Function .

2. **ReSource Filter :**

Iresource Filter is used ….Attribute

Example Code :

```
 public class MySampleResourceFilter :Attribute, IResourceFilter
 {
    private readonly string  _name;
    public MySampleResourceFilter(string name="Global")
    {
       _name = name;
    }
    public void  OnResourceExecuted(ResourceExecutedContext context)
    {
       Console.WriteLine($"Resource Filter - After {_name} ");
       //throw new NotImplementedException();
    }

    public void OnResourceExecuting(ResourceExecutingContext context)
    {
       Console.WriteLine($"Resource Filter - Before {_name} ");
      // throw new NotImplementedException();
    }
 }
```

5 .Result Filters :

Result Filters runs before and after the result of the action method is executed .
IResult Filter , IAlwaysRunResultFilter
Example :

```
[LogResultFilter]
Public ActionResult Index(){

        Return View();
}
Public class LogResultFilter :FilterAttribute, IResultFilter {
        Public void OnResultExecuting(ResultExecutingContext filters){
                        //This code
        }
        Public void OnResultExecuted(ResultExecutedContext filters){
                //
        }
}
```

//Result filters runs before and after the result of action method is executed .

4 .Actions Filters :

Action Filters are custom filters which can run before and after an action method is executed .

```
[LogActionFilter]
Public ActionResult Index(){

}
Public class LogActionFilter : ActionFilterAttribute
{
        Public override void OnActionExecuting(){

        }
        Public override void OnActionExecuted(){

        }
}
```

Dependency Injunction In Filters :

Filters can be added by type or by instance. If an instance is added, that instance is used for every request.

- ServiceFilterAttribute
- TypeFilterAttribute

For Declaring Globally :
options.Filters.AddService<MySampleResultAttribute>();

And also builder.Services.AddTransient<MySampleResultAttribute>();

Service Filters and Type Filters are ,more are less same for executing DI in from controller, action methods :

```
[MySampleResourceFilter("Resource")]
[ServiceFilter(typeof(MySampleResultAttribute))]
[TypeFilter(typeof(MySampleResultAttribute),Arguments=new object[] {"Action"})]
[MySampleActionFilterAttribute("Action",1)]
```

HTML Helpers , Model Binding

Wednesday, January 24, 2024 4:53 PM

VIEWS

Tuesday, January 23, 2024 4:35 PM

@model IEnumerable<Players>//must be added for that Because It returns the list of players details .

Index .CSHTML :

```
@model IEnumerable<Product>

<h2>Product List</h2>

<table class="table">
  <thead>
    <tr>
      <th>Id</th>
      <th>Name</th>
      <th>Price</th>
      <th>Actions</th>
    </tr>
  </thead>
  <tbody>
    @foreach (var product in Model)
    {
      <tr>
        <td>@product.Id</td>
        <td>@product.Name</td>
        <td>@product.Price</td>
        <td>
          <a asp-action="Details" asp-route-id="@product.Id">Details</a> |
          <a asp-action="Edit" asp-route-id="@product.Id">Edit</a> |
          <a asp-action="Delete" asp-route-id="@product.Id">Delete</a>
        </td>
      </tr>
    }
  </tbody>
</table>
```

Details.CSHTML :

```
@model Product

<h2>Product Details</h2>

<div>
  <dl class="row">
    <dt class="col-sm-2">Id</dt>
    <dd class="col-sm-10">@Model.Id</dd>

    <dt class="col-sm-2">Name</dt>
    <dd class="col-sm-10">@Model.Name</dd>
```

```
        <dt class="col-sm-2">Price</dt>
        <dd class="col-sm-10">@Model.Price</dd>
    </dl>
</div>

<p>
    <a asp-action="Index">Back to List</a>
</p>
```

Create.CSHTML

```
@model Product

<h2>Create Product</h2>

<form asp-action="Create" method="post">
    <div class="form-group">
        <label asp-for="Name"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label asp-for="Price"></label>
        <input asp-for="Price" class="form-control" />
        <span asp-validation-for="Price" class="text-danger"></span>
    </div>

    <button type="submit" class="btn btn-primary">Create</button>
    <a asp-action="Index">Back to List</a>
</form>
```

Edit.CSHTML :

```
@model Product

<h2>Edit Product</h2>

<form asp-action="Edit" method="post">
    <input type="hidden" asp-for="Id" />

    <div class="form-group">
        <label asp-for="Name"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label asp-for="Price"></label>
        <input asp-for="Price" class="form-control" />
        <span asp-validation-for="Price" class="text-danger"></span>
```

```
    </div>

    <button type="submit" class="btn btn-primary">Save Changes</button>
    <a asp-action="Index">Back to List</a>
</form>
```

Delete.CSHTML :

```
@model Product

<h2>Delete Product</h2>

<p>Are you sure you want to delete this product?</p>

<dl class="row">
    <dt class="col-sm-2">Id</dt>
    <dd class="col-sm-10">@Model.Id</dd>

    <dt class="col-sm-2">Name</dt>
    <dd class="col-sm-10">@Model.Name</dd>

    <dt class="col-sm-2">Price</dt>
    <dd class="col-sm-10">@Model.Price</dd>
</dl>

<form asp-action="ConfirmDelete" method="post">
    <input type="hidden" asp-for="Id" />
    <button type="submit" class="btn btn-danger">Delete</button>
    <a asp-action="Index">Back to List</a>
</form>
```

ASP-ROUTE-ACTION:
It goes to that action page
ASP-ROUTE-ID :
It goes to that Id only in that URL

Attribute Routing :

```
//For that Route ConfirmDelete is the action .
 [HttpPost]
 [Route("Player/Delete")]
 public IActionResult ConfirmDelete(int id)
 {
    var product = _sqlRepository.GetById(id);

    if (product == null)
    {
       return NotFound();
    }

    _sqlRepository.Delete(product);
    return RedirectToAction("Index");
```

```
}
```

Here I used the name of Confirm Delete , but in returning there is no delete action . So that I attributed that routing in to that .

ASP-for, asp-validation for

Tuesday, January 23, 2024 5:37 PM

Asp-for="Name" for that field of that class data.

As-validation-for="Name" It validates that data and returns error if any error

Creating Model with Basic Validations

Tuesday, January 23, 2024 12:28 PM

```
[Required(ErrorMessage = "Category is required")]
    public cat Category { get; set; }\
```

Explanation:

This line uses the Required attribute to specify that the Category property must have a value. If it doesn't, it will trigger a validation error with the specified error message ("Category is required").

```
[Required(ErrorMessage = "BookName is required")]
[RegularExpression("^[a-zA-Z ]+$", ErrorMessage = "BookName can only contain letters")]
public string BookName { get; set; }
```

Explanation :

- This line specifies that the BookName property is required, and if it's not provided, a validation error with the message "BookName is required" will be triggered.
- Additionally, a regular expression is used to enforce that BookName should only contain letters (uppercase or lowercase) and spaces. If the regular expression is not matched, a validation error with the specified message will be triggered.

```
//[Required(ErrorMessage = "Author is required")]
[RegularExpression("^[a-zA-Z ]+$", ErrorMessage = "Author can only contain letters")]
public string Author { get; set; }
```

Explanation :

- The Author property is defined with a regular expression constraint similar to BookName. It allows only letters (uppercase or lowercase) and spaces.
- Unlike BookName, the Author property is not marked as required, so it's optional. If you want to make it required, you can uncomment the [Required] attribute.

```
[Range(1000, 9999, ErrorMessage = "Published year must be between 1000 and 9999")]
public int? Published { get; set; }
```

Explanation :

- The Published property is decorated with the Range attribute, which ensures that the value falls within the specified range. In this case, the range is set between 1000 and 9999.
- If the value of Published is outside this range, a validation error with the specified message ("Published year must be between 1000 and 9999") will be triggered

Int ?Means :

- int: This is the standard integer type in C#. It can hold whole numbers without decimal points.
- int?: This is the nullable version of int. It can hold any integer value or be assigned null.

```
[Range(0.01, double.MaxValue, ErrorMessage = "Price must be greater than 0")]
public double? Price { get; set; }
```

- Similar to Published, the Price property is decorated with the Range attribute. It ensures that the value of Price falls within the specified range (greater than 0).
- If the value is less than or equal to 0, a validation error with the specified message ("Price must be greater than 0") will be triggered.

```
public string Updated { get; set; }
```

The Updated property is a string property, and it doesn't have any validation

Add SingleTon, Add Scoped , Add Transcient

Tuesday, January 23, 2024 11:58 AM

Add SingleTon :

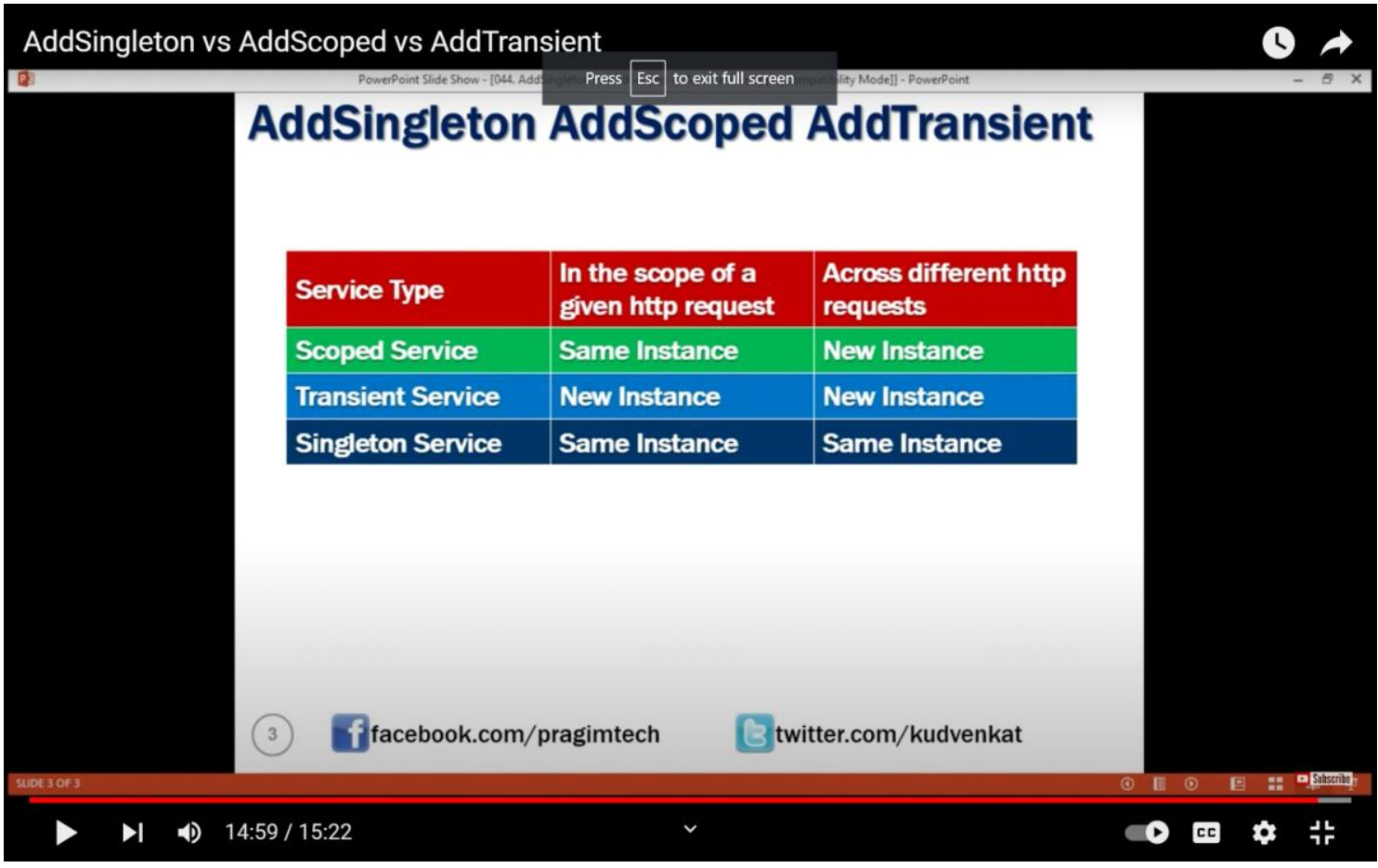
The SingletonUserName in the output should be consistent across multiple requests because AddSingleton creates a single instance of the service

Add Scoped :

It did n't allowed the duplicates of that values .
Usually, we always using this scoped methods .

Add Transcient :

The TransientUserName in the output should change on each request because AddTransient creates a new instance of the service



IRepositoryPattern

Tuesday, January 23, 2024 11:54 AM

In the Irepository Pattern , we can create many repositoruies for that Interface .

EXAMPLE :

- The Product class represents the entity you want to manage.
- The IRepository interface defines the contract for a generic repository.
- The ProductRepository class implements the repository interface, providing concrete implementations for CRUD operations.
- The AppDbContext class represents the DbContext for Entity Framework Core.
- The Startup.cs file configures the DbContext and registers the repository as a scoped service.
- The ProductController demonstrates how to use the repository in a controller.

Product.CS Class

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    // Other properties...
}
```

Irepository :

```
public interface IRepository<TEntity> where TEntity : class
{
    TEntity GetById(int id);
    IEnumerable<TEntity> GetAll();
    void Add(TEntity entity);
    void Update(TEntity entity);
    void Delete(TEntity entity);
}
```

Repository Implementation :

```
public class ProductRepository : IRepository<Product>
{
    private readonly DbContext dbContext;

    public ProductRepository(DbContext dbContext)
    {
        this.dbContext = dbContext;
    }

    public Product GetById(int id)
    {
        return dbContext.Set<Product>().Find(id);
```

```
    }

    public IEnumerable<Product> GetAll()
    {
        return dbContext.Set<Product>().ToList();
    }

    public void Add(Product entity)
    {
        dbContext.Set<Product>().Add(entity);
        dbContext.SaveChanges();
    }

    public void Update(Product entity)
    {
        dbContext.Set<Product>().Update(entity);
        dbContext.SaveChanges();
    }

    public void Delete(Product entity)
    {
        dbContext.Set<Product>().Remove(entity);
        dbContext.SaveChanges();
    }
}
```

DBContext (AppDbContext) :

```
public class AppDbContext : DbContext
{
    public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)
    {
    }

    public DbSet<Product> Products { get; set; }
    // Other DbSets...
}
```

DependencyInjucntion Configuration In Program.CS :

```
 services.AddScoped<IRepository<Product>, ProductRepository>();
```

Usage of Controller :

```
public class ProductController : Controller
{
    private readonly IRepository<Product> productRepository;

    public ProductController(IRepository<Product> productRepository)
    {
        this.productRepository = productRepository;
    }
```

```
public IActionResult Index()
{
   var products = productRepository.GetAll();
   return View(products);
}

public IActionResult Details(int id)
{
   var product = productRepository.GetById(id);

   if (product == null)
   {
      return NotFound();
   }

   return View(product);
}

public IActionResult Create()
{
   return View();
}

[HttpPost]
public IActionResult Create(Product product)
{
   if (ModelState.IsValid)
   {
      productRepository.Add(product);
      return RedirectToAction("Index");
   }

   return View(product);
}

public IActionResult Edit(int id)
{
   var product = productRepository.GetById(id);

   if (product == null)
   {
      return NotFound();
   }

   return View(product);
}

[HttpPost]
public IActionResult Edit(int id, Product product)
{
   if (id != product.Id)
   {
```

```
            return BadRequest();
        }

        if (ModelState.IsValid)
        {
            productRepository.Update(product);
            return RedirectToAction("Index");
        }

        return View(product);
    }

    public IActionResult Delete(int id)
    {
        var product = productRepository.GetById(id);

        if (product == null)
        {
            return NotFound();
        }

        return View(product);
    }

    [HttpPost, ActionName("Delete")]
    public IActionResult ConfirmDelete(int id)
    {
        var product = productRepository.GetById(id);

        if (product == null)
        {
            return NotFound();
        }

        productRepository.Delete(product);
        return RedirectToAction("Index");
    }
}
```

Creating Views For That :

Index.CSHTML :

```
@model IEnumerable<Product>

<h2>Product List</h2>

<table class="table">
    <thead>
        <tr>
            <th>Id</th>
            <th>Name</th>
```

```
        <th>Price</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
      @foreach (var product in Model)
      {
        <tr>
          <td>@product.Id</td>
          <td>@product.Name</td>
          <td>@product.Price</td>
          <td>
            <a asp-action="Details" asp-route-id="@product.Id">Details</a> |
            <a asp-action="Edit" asp-route-id="@product.Id">Edit</a> |
            <a asp-action="Delete" asp-route-id="@product.Id">Delete</a>
          </td>
        </tr>
      }
    </tbody>
</table>
```

Details.CSHTML :

```
@model Product

<h2>Product Details</h2>

<div>
    <dl class="row">
        <dt class="col-sm-2">Id</dt>
        <dd class="col-sm-10">@Model.Id</dd>

        <dt class="col-sm-2">Name</dt>
        <dd class="col-sm-10">@Model.Name</dd>

        <dt class="col-sm-2">Price</dt>
        <dd class="col-sm-10">@Model.Price</dd>
    </dl>
</div>

<p>
    <a asp-action="Index">Back to List</a>
</p>
```

Create.CSHTML

```
@model Product

<h2>Create Product</h2>

<form asp-action="Create" method="post">
    <div class="form-group">
        <label asp-for="Name"></label>
```

```
    <input asp-for="Name" class="form-control" />
    <span asp-validation-for="Name" class="text-danger"></span>
  </div>

  <div class="form-group">
    <label asp-for="Price"></label>
    <input asp-for="Price" class="form-control" />
    <span asp-validation-for="Price" class="text-danger"></span>
  </div>

  <button type="submit" class="btn btn-primary">Create</button>
  <a asp-action="Index">Back to List</a>
</form>
```

Edit.CSHTML :

```
@model Product

<h2>Edit Product</h2>

<form asp-action="Edit" method="post">
  <input type="hidden" asp-for="Id" />

  <div class="form-group">
    <label asp-for="Name"></label>
    <input asp-for="Name" class="form-control" />
    <span asp-validation-for="Name" class="text-danger"></span>
  </div>

  <div class="form-group">
    <label asp-for="Price"></label>
    <input asp-for="Price" class="form-control" />
    <span asp-validation-for="Price" class="text-danger"></span>
  </div>

  <button type="submit" class="btn btn-primary">Save Changes</button>
  <a asp-action="Index">Back to List</a>
</form>
```

Delete.CSHTML :

```
@model Product

<h2>Delete Product</h2>

<p>Are you sure you want to delete this product?</p>

<dl class="row">
  <dt class="col-sm-2">Id</dt>
  <dd class="col-sm-10">@Model.Id</dd>

  <dt class="col-sm-2">Name</dt>
  <dd class="col-sm-10">@Model.Name</dd>
```

```
    <dt class="col-sm-2">Price</dt>
    <dd class="col-sm-10">@Model.Price</dd>
</dl>

<form asp-action="ConfirmDelete" method="post">
    <input type="hidden" asp-for="Id" />
    <button type="submit" class="btn btn-danger">Delete</button>
    <a asp-action="Index">Back to List</a>
</form>
```

ScaffOld Project

Tuesday, January 23, 2024 11:35 AM

In scoffold Project, it will automatically generate views, controllers,DbContext,we can be used to combine database to the Controller(Project) .

DB First Approaches (With Web API)

Monday, January 22, 2024 11:40 AM

Create a new File using asp.net core web api mvc

Install NuGet Packages :
dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL
dotnet add package Microsoft.EntityFrameworkCore.Design

Scaffold The DataContext :
Find the Directory:
1 .So go to the package manager console ==> "dir" and check
2 .Cd solutionName
Now we are in the location of that project

3.Command Line :

dotnet ef dbcontext scaffold
"Host=YourHost;Database=YourDatabase;Username=YourUsername;Password=YourPassword"
Npgsql.EntityFrameworkCore.PostgreSQL -o Models -c YourDbContext

4. Builder.Services.AddDbContext In the program.cs
   builder.Services.AddDbContext<MyDbContext>(options =>
   { options.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection")); });
   Now Models of that DataBase and That DbContext in that name of MyDbContext . It will Create automatically because it is scaffolding .

   5.Creating a Controller :

   -->private fieeld with Constructor for DbContext

   Get :
   Example :

```
[HttpGet("Get Books")]
public async Task<ActionResult<List<Book>>> GetBooks()
{
    return Ok(await _context.Books.ToListAsync());
}
```

   6 .Getting One Information Example :

```
[HttpGet("GetSingleTeacherDetails")]
public async Task<ActionResult<List<Teachers>>> GetSingleTeacher(int id)
{


   var FindTeacher = await _context.MyDbSetTeachers.FindAsync(id);
   if(FindTeacher == null)
```

```
        {
            return BadRequest("Invalid Data");
        }

        return Ok(FindTeacher);
    }
```

7.HTTP Post Addding

Example :

```
[HttpPost("AddTeachers")]
public async Task<ActionResult<List<Teachers>>> AddTeacher(Teachers request)
{
    _context.MyDbSetTeachers.Add(request);
    await _context.SaveChangesAsync();
    return Ok(await _context.MyDbSetTeachers.ToListAsync());
}
```

8 . Delete From that DataBase :

Example :

```
[HttpDelete("DeleteTeachers")]
public async Task<ActionResult<List<Teachers>>> DeleteTeacher(int id)
{

    var dBFindTeacher = await _context.MyDbSetTeachers.FindAsync(id);
    if (dBFindTeacher == null)
    {
        return BadRequest("Invalid Data");
    }
    _context.Remove(dBFindTeacher);
    await _context.SaveChangesAsync();
    return Ok(await _context.MyDbSetTeachers.ToListAsync());
}
```

9. Put into that DataBase :

```
[HttpPut("ModifyTeachers")]
public async Task<ActionResult<List<Teachers>>> ModifyTeacher(int id,Teachers request)
{

   var dBFindTeacher = await _context.MyDbSetTeachers.FindAsync(id);
   if (dBFindTeacher == null)
   {
      return BadRequest("Invalid Data");
   }
  dBFindTeacher.TeacherId=id;
   dBFindTeacher.TeacherName= request.TeacherName;
   dBFindTeacher.LastName= request.LastName;
   dBFindTeacher.Class=request.Class;
   await _context.SaveChangesAsync();
```

```
    return Ok(await _context.MyDbSetTeachers.ToListAsync());
}
```

Assignments

Wednesday, January 17, 2024 5:48 PM

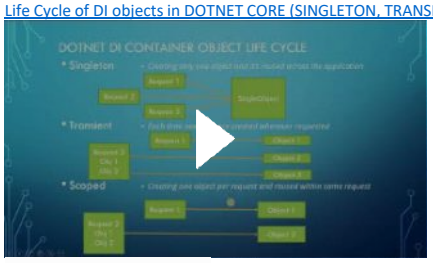
Api Basics	

Dependency Injunction

Friday, January 19, 2024 10:51 AM

Dependency Injunction a Class downs not create an instance of another class. Instead of , it will ask others to create .

Life Cycle of DI objects in DOTNET CORE (SINGLETON, TRANSIENT, SCOPED) - Tamil

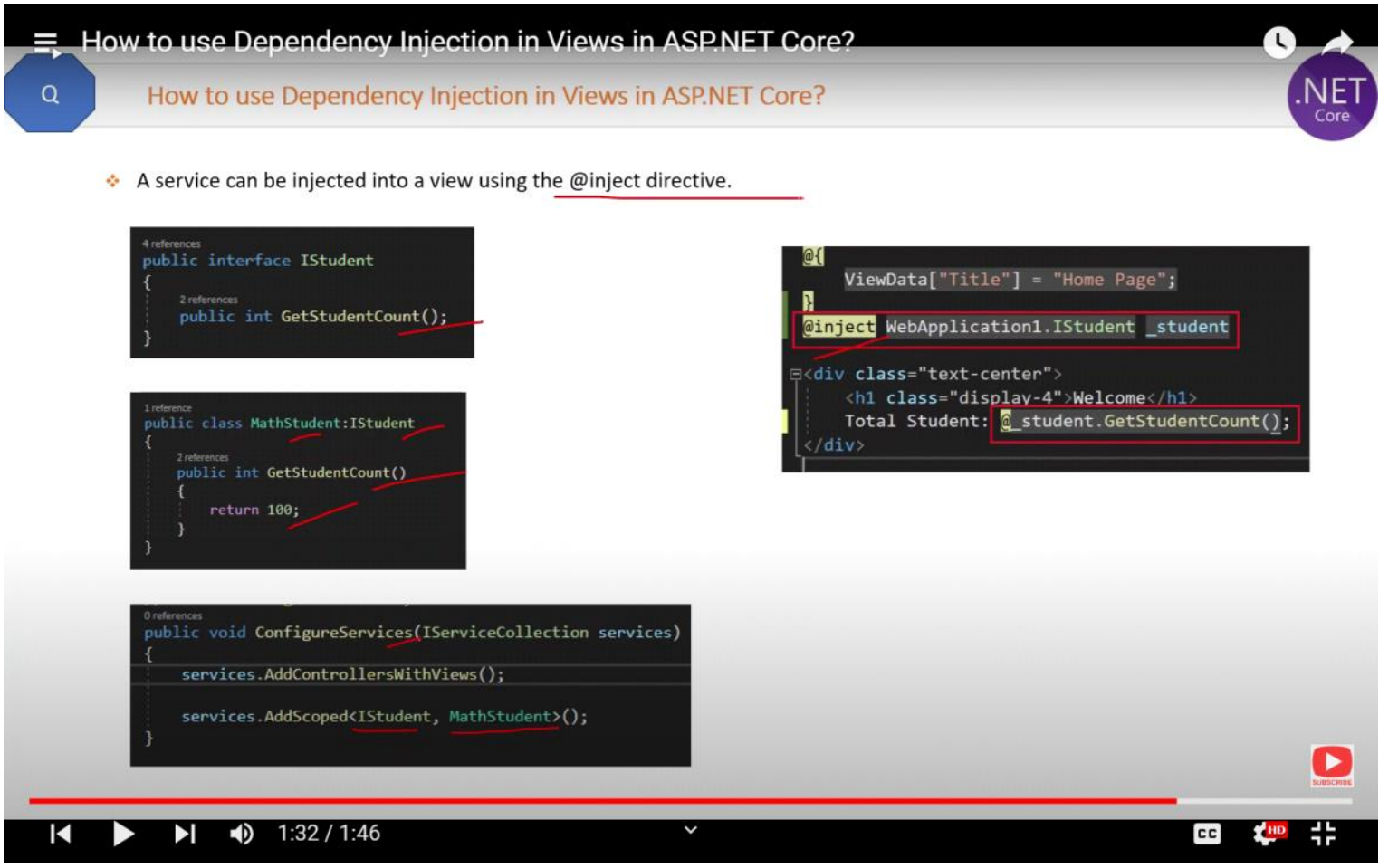


Dependency Injection is the design pattern that helps us to create an application which loosely coupled.

greater maintainability, testability, and also re-usability

Dependency Injunction via Controller :

```
public HomeController(IHelloWorldService helloWorldService)
{
_helloWorldService = helloWorldService;
}
```



Inject the dependency in the controller action :

```
public IActionResult Index ( [FromServices] IHelloWorldService helloWorldService ) {

ViewData["MyText"] = helloWorldService.SaysHello() + "Jignesh!"; return View();

}
```

Get the service instance manually :

```
public IActionResult Index1() { var helloWorldService = (IHelloWorldService)
this.HttpContext.RequestServices.GetService(typeof(IHelloWorldService)); ViewData["MyText"] =
```

helloWorldService.SaysHello() + "Jignesh Trivedi!"; return View("index"); }

DI Containers:

A DI Container is a framework to creates dependencies and injects them automatically when required. It automatically creates objects based on the request and injects them when required. DI Container helps us to manage dependencies within the application simply and easily.

LifeCycle:

1.Register
2.Resolve
3.Dispose

Rule :

High level module (class) should not depend on the Low level module or class . Both should depend on abstraction .
===> should not depend (directly) .
===>depend only on abstraction .
BY using the field we can implements into that .
constructor injection, method injection, and property injection
Constructor Injunction :
Constructor Injection is the act of statically defining the list of required Dependencies by specifying them as parameters to the class's constructor.
Method Injunction :
dependency injection is a technique whereby one object (or static method) supplies the dependencies of another object
Property injunction:
type of dependency injection where dependencies are provided through properties

Code :

```
builder.Services.AddScoped<IEmployee, EmployeeRepository>();
```

.NET's dependency injection there are three major lifetimes:

Singleton which creates a single instance throughout the application. It creates the instance for the first time and reuses the same object in the all calls.

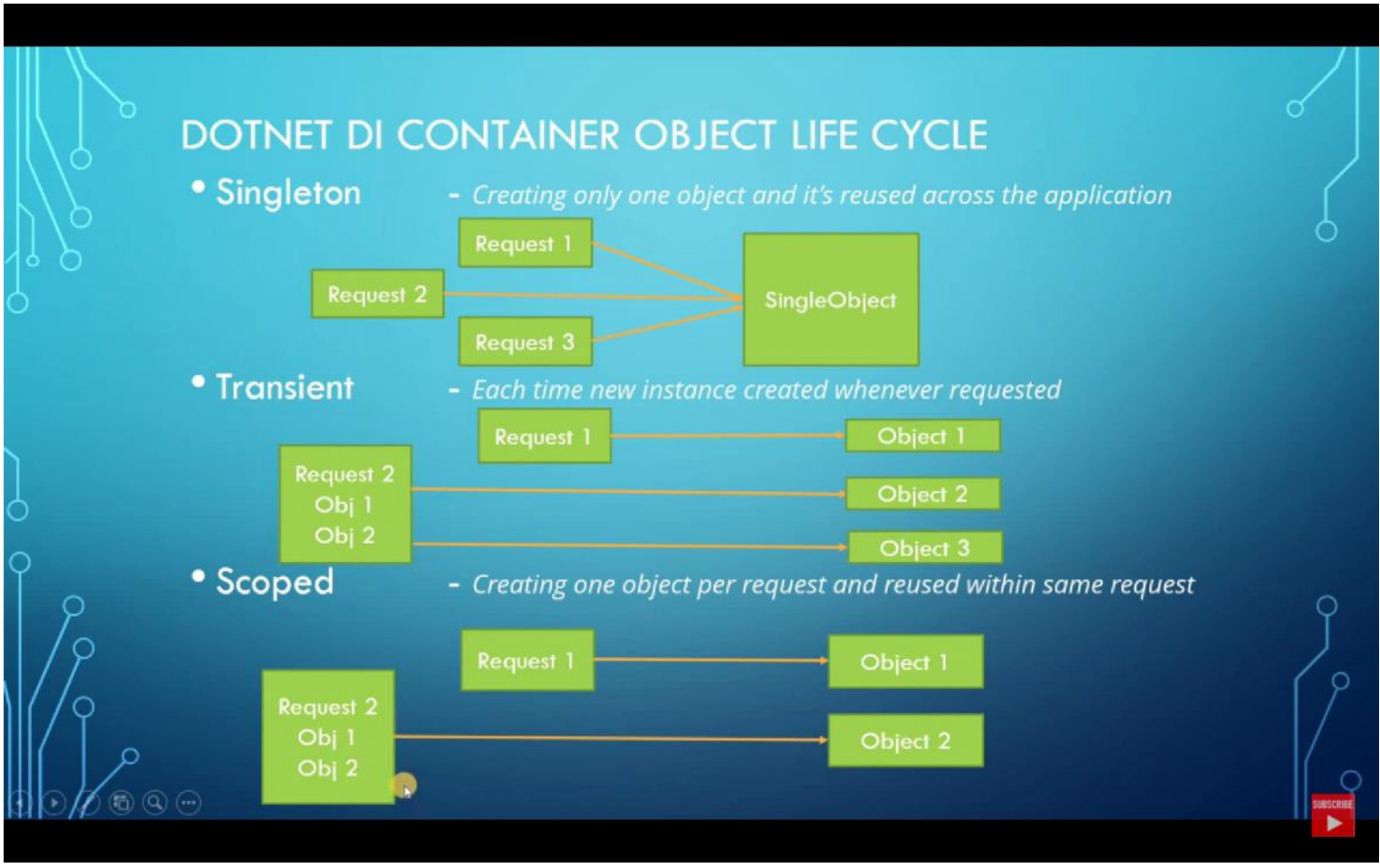
a Singleton service is created only one time per application and that single instance is used throughout the application life time.

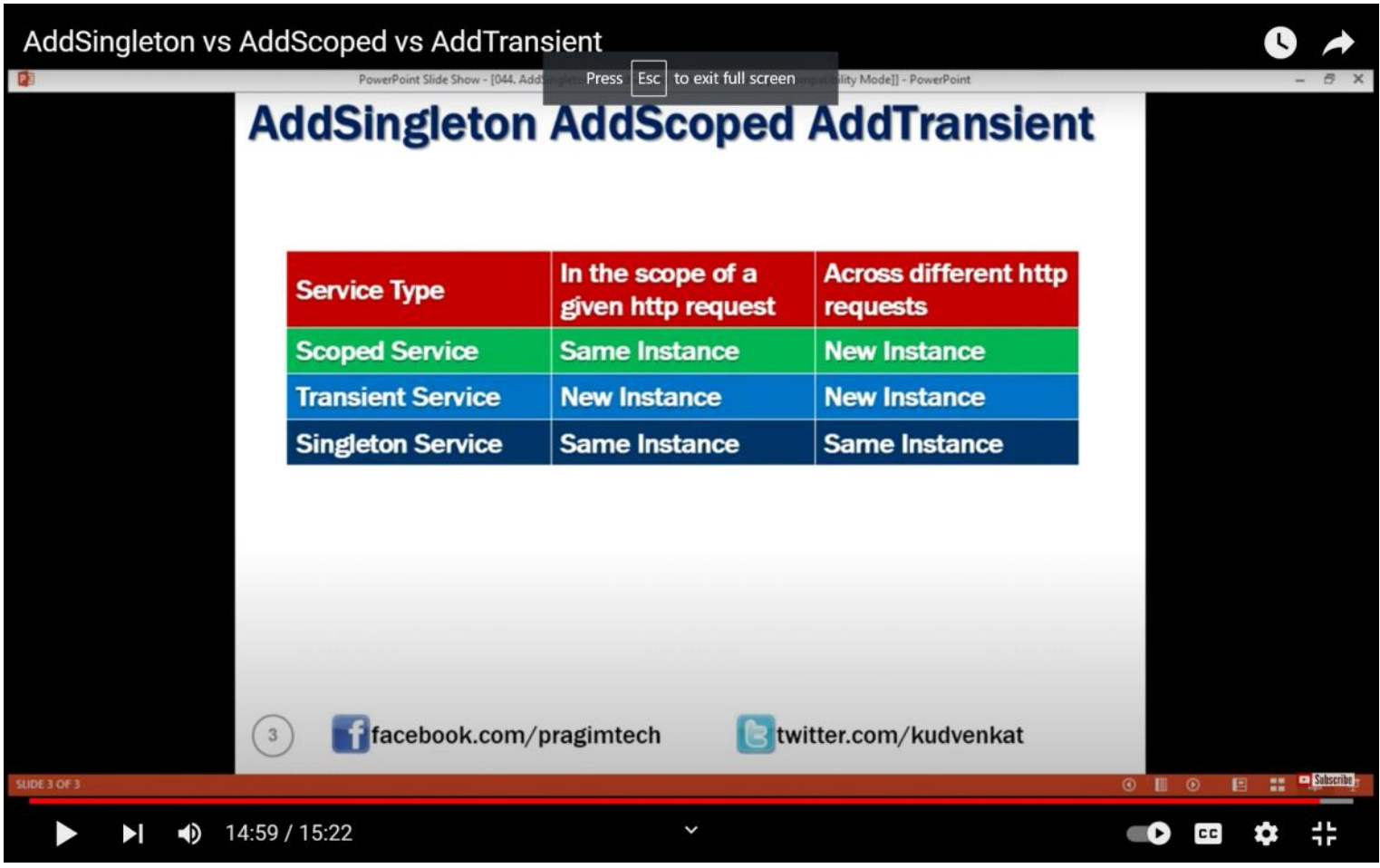
- memory leaks in these services will build up over time.
- also memory efficient as they are created once reused everywhere.

Scoped lifetime services are created once per request within the scope. It is equivalent to a singleton in the current scope. For example, in MVC it creates one instance for each HTTP request, but it uses the same instance in the other calls within the same web request.

Transient lifetime services are created each time they are requested. This lifetime works best for lightweight, stateless services.

- since they are created every time they will use **more memory** & Resources and can have a **negative** impact on performance
- use this for the **lightweight** service with little or **no state**.


DOTNET DI CONTAINER OBJECT LIFE CYCLE
• Singleton
- Creating only one object and it's reused across the application
Request 1
Request 2
Request 3
SingleObject
• Transient
- Each time new instance created whenever requested
Request 1
Object 1
Request 2
Obj 1
Obj 2
Object 2
Object 3
• Scoped
- Creating one object per request and reused within same request
Request 1
Object 1
Request 2
Obj 1
Obj 2
Object 2



Precedence In Dependency Injunction :

If suppose you give three lifetimes in a same project and what will be workings first ? Ovverride will be happening . The last thing will be happening first . Because first one is overrided . After that second one is overrided by third One .

WebApi and Swaggers Example 2

Thursday, January 18, 2024 3:15 PM

Package Manager Console To Update in the Database :

```
dotnet ef migrations add InitialCreate
dotnet ef database update
```

Inplementing In The Program.Cs

```
builder.Services.AddDbContext<DataContext>(options =>
{ options.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection")); });
```

In DataContext Page Creation :

```
using TeachersApi.Model;
namespace TeachersApi.Data
{
   public class DataContext:DbContext
   {

      public DataContext(DbContextOptions<DataContext> options) : base(options) { }
      public DbSet<Teachers> MyDbSetTeachers { get; set; }
   }
}
```

DataAnnotations :

Key
Required

Async and Await :

Synchronous -> Even if threads are low, the works are in low perfomance .
Asynchronous->Even if the threads are high , the works are in high performance .

```
Public async Task<return Type >(){
        Return await
}
```

Async-->Task-->Await .In the keyword of await ,it will become asynchronous .
For Multi Tasking and Work load Management .

Basic Validations :

Range(1,100)-->Values like orders
[MaxLength(30)]-->Display Name

```
<span asp-validation-for="Name" class="text-danger" ></span>
```

Connection String in the JSON Files with PostGres Sql:

```
  "ConnectionStrings": {
    "DefaultConnection": "Server=34.23.55.201;DataBase=training-db;UserName=freshers-
training;Password=kV1O1st2Uukp9=2"
  },
```

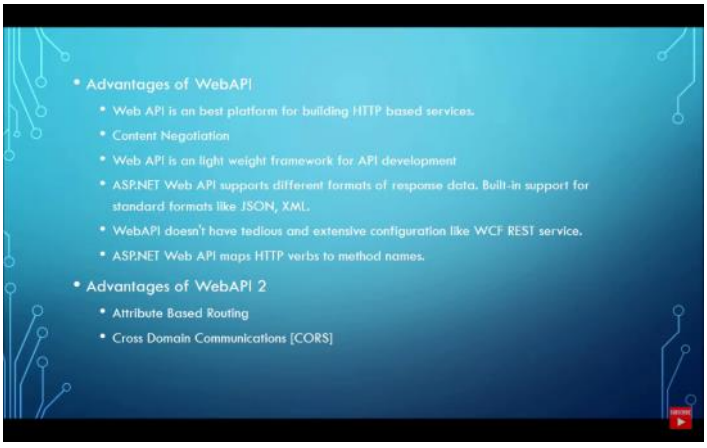
GivingRouterName:

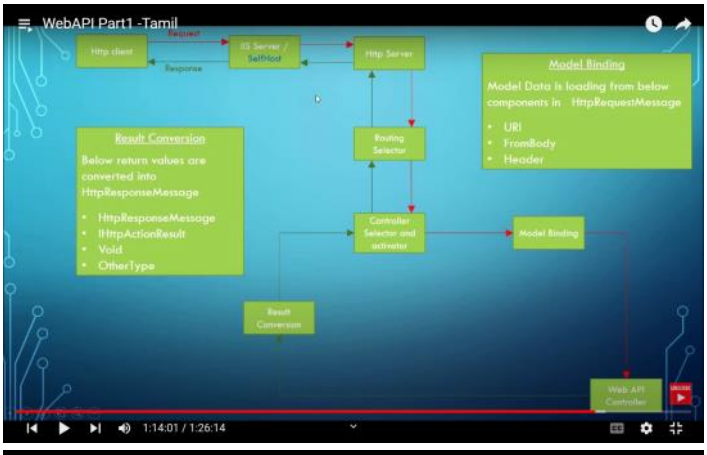
```
[HttpGet("GetTeachers")]
```

Assignments

Thursday, January 18, 2024 3:28 PM

Api Basics	 TeachersApi
	 TeachersApiWithDb
Controller and I repository	 IRepository
Db First Example	 DbFirstApproachExa...

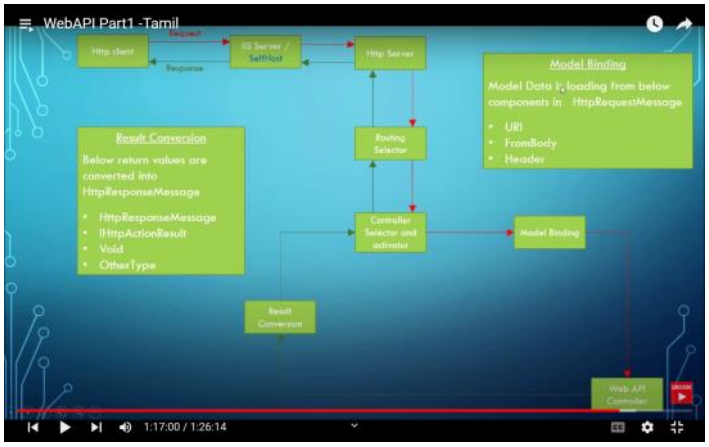

• Advantages of WebAPI
• Web API is an best platform for building HTTP based services.
• Content Negotiation
• Web API is an light weight framework for API development
• ASP.NET Web API supports different formats of response data. Built-in support for standard formats like JSON, XML.
• WebAPI doesn't have tedious and extensive configuration like WCF REST service.
• ASP.NET Web API maps HTTP verbs to method names.
• Advantages of WebAPI 2
• Attribute Based Routing
• Cross Domain Communications [CORS]

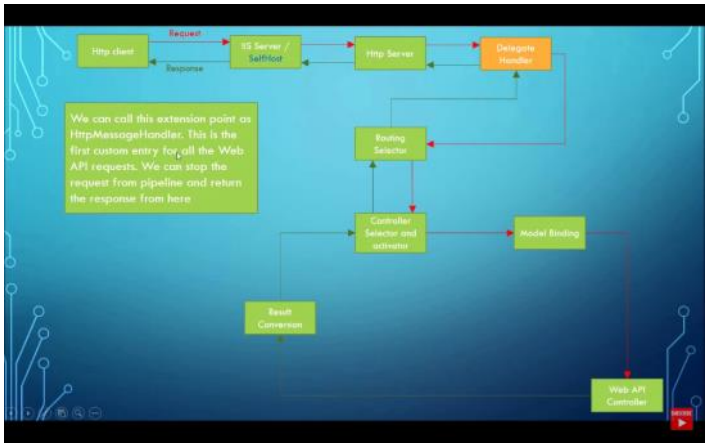

WebAPI Part1 -Tamil
Http client
Request
Response
IIS Server / Selfhost
Http Server
Routing Selector
Controller Selector and activator
Model Binding
Result Conversion
Web API Controller
Model Binding
Model Data is loading from below components in HttpRequestMessage
• URI
• FromBody
• Header
Result Conversion
Below return values are converted into HttpResponseMessage
• HttpResponseMessage
• IHttpActionResult
• Void
• OtherType
1:14:01 / 1:26:14

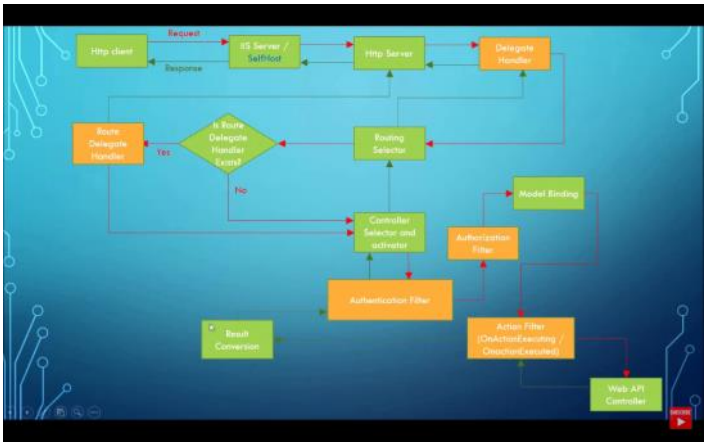

WebAPI Part1 -Tamil
CUSTOM EXTENSION
• Global Exception Handling
• Authentication / Authorization
• Logging
• Common Logic for all controllers (or set of controller)
Etc....
1:16:49 / 1:26:14

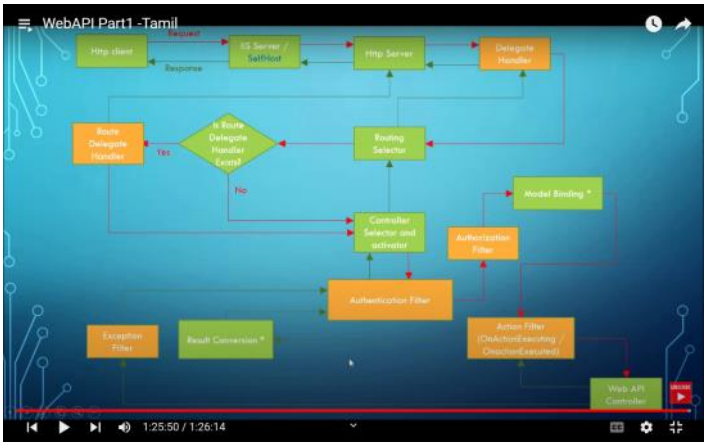
BEFORE THE ROTING POIINT WILL BE CAME , WE CAN MAKE OUR OWN AUTHENTICATION AND OTHER MIDDLEWARES KIND OF THINGS .

- Like Jio Restricted from Japan , we don't need to do after routing . It is not effective and also not efficient . So we can use it before the efficient part is before the routing all the middle wares and custom filters authentication and authorization kind of things .









1. **HttpRequest Message: This is like the letter you write to your friend. In your letter, you say, "Please send me a picture of your new puppy." You put your letter in an envelope, write your friend's address on it, and give it to the mailman. This is like when you use a computer or a phone to ask for something on the internet, like looking at a website or a picture. The HttpRequest is your way of telling the internet what you want to see or get.**

2. **HttpResponse Message: Now, think about when your friend gets your letter. They read it and understand that you want a picture of their puppy. So, they take a nice photo, put it in an envelope, write your address on it, and send it back to you. When the mailman brings it to your house, you open the envelope and see the picture! That's like the HttpResponse. It's the internet's way of answering your request and giving you what you asked for, like showing you a website or a picture.**

3 . CANCELLATION TOKEN :

Okay! Imagine you're playing a game where you're building a big tower with blocks. You keep adding more and more blocks to make it taller. But suddenly, your mom calls you for dinner. So, what do you do? You stop building the tower and go to eat dinner, right?
In computer world, when a program is doing a task, like counting numbers or loading a picture, it's like building a tower with blocks. Sometimes, the computer needs to stop doing that task because maybe it's taking too long, or you want to do something else. A cancellation token is like your mom calling you for dinner. It's a way for the computer to say, "Hey, stop doing that task and do something else!"
So, a cancellation token is like a signal that can tell a computer program to stop doing a task. This way, the computer can do other important things instead, just like how you stop building your tower to go eat dinner!

StatelessNess :

Alright, let's think about playing a game of tag. When you're "It," you try to tag someone else, and then they become "It." But imagine if every time you played, you had to remember who was "It" last week, or the week before, or even last year. That would be really hard, right?
Statelessness in computers is kind of like playing tag without having to remember who was "It" in all the past games. Every time you play, it's like a brand new game. When you use a computer or a phone to look at websites or use apps, the internet works in a similar way. Each time you ask to see a website (like asking for your favorite cartoon), the internet shows it to you without needing to remember what you did last time or the time before.
So, statelessness means that every time you ask for something on the internet, it's like starting fresh, without the need to remember your past requests or actions. Just like playing a new game of tag every time, where it doesn't matter who was "It" before!

END POINTS URLS :

Alright, let's imagine you're in a big library full of books. Each book is about something different, like dinosaurs, space, or fairy tales. Now, imagine if each book had a special spot where it always sits on the shelf. To find the book you want, you just need to know its special spot.
Endpoint URLs are like those special spots in the library for books. When you use a computer, tablet, or phone to look at websites or ask for information on the internet, each piece of information has its own special spot, just like books in the library. These special spots are called endpoint

URLs.

LinQ

Wednesday, January 24, 2024 7:42 PM

QUANTIFIERS IN LINQ

Tuesday, February 6, 2024 2:52 PM

Quantifiers in LINQ are methods that return a boolean value indicating whether some or all elements in a sequence satisfy a given condition. These methods are powerful tools for checking the presence or absence of certain conditions within collections. The primary quantifier operations in LINQ include Any, All, and Contains.

1. Any

The Any operator checks if any elements in a sequence satisfy a specified condition. If at least one element meets the condition, it returns true; otherwise, it returns false. You can also use Any without a condition to determine if the sequence contains any elements at all.

```
var numbers = new[] { 1, 2, 3, 4, 5 };

bool hasEven = numbers.Any(n => n % 2 == 0); // true, because there are even numbers
bool hasGreaterThanTen = numbers.Any(n => n > 10); // false, no number is greater than 10
```

2. All

The All operator is used to verify that all elements in a sequence satisfy a specified condition. If every element meets the condition, All returns true. If any element does not meet the condition, it returns false.

```
bool allEven = numbers.All(n => n % 2 == 0); // false, because not all numbers are even
bool allLessThanTen = numbers.All(n => n < 10); // true, all numbers are less than 10
```

3. Contains

The Contains operator determines whether a sequence contains a specific element. If the element is found in the sequence, Contains returns true; otherwise, it returns false. This method can use an optional IEqualityComparer<T> to determine equality.

```
bool containsThree = numbers.Contains(3); // true, because the sequence includes 3
```

Sequence equal Operator

Tuesday, February 6, 2024 2:49 PM

The SequenceEqual operator in LINQ is used to determine whether two sequences are equal in terms of both their length and the elements they contain, where each element in one sequence is equal to the corresponding element in the other sequence, and in the same order. This is a strict comparison that goes beyond just checking if the two sequences contain the same elements; it also verifies that the elements appear in the same order.

```
int[] sequence1 = { 1, 2, 3, 4 };
int[] sequence2 = { 1, 2, 3, 4 };
int[] sequence3 = { 4, 3, 2, 1 };

bool isEqual1 = sequence1.SequenceEqual(sequence2); // true
bool isEqual2 = sequence1.SequenceEqual(sequence3); // false
```

Concat Operator

Tuesday, February 6, 2024 2:46 PM

The Concat operator in LINQ is used to append one sequence to another. It combines two sequences end-to-end, creating a new sequence that contains all the elements from the first sequence followed by all the elements from the second sequence, in their original order. This operator is useful when you need to merge data from two sources or simply want to create a longer sequence from two shorter ones.

```
int[] first = { 1, 2, 3 };
int[] second = { 4, 5, 6 };

var combined = first.Concat(second);
```

GENERATION Operators

Tuesday, February 6, 2024 2:37 PM

Range
Repeat
Empty

Generation operators in LINQ are used to create new sequences of elements. These operators are handy when you need to instantiate a collection in a concise way or generate a sequence of numbers that fits a certain criteria. The most commonly used generation operators include Empty, Repeat, and Range.

1. Empty

The Empty operator creates an empty sequence of a specified type. It's useful when you need a sequence that has no elements but should be of a specific type, often used to ensure the return of a method is not null but instead an empty collection.

```
var emptySequence = Enumerable.Empty<int>();
```

2. Repeat

The Repeat operator generates a sequence that contains a repeated value a specified number of times. This can be useful when you need a collection with the same value repeated multiple times.

```
var repeatedSequence = Enumerable.Repeat("Hello", 3);
// Result: "Hello", "Hello", "Hello"
```

Range

The Range operator generates a sequence of numbers within a specified range, where you provide the starting number and the count of sequential numbers to generate. It's particularly useful for generating a sequence of integers.

```
var numberSequence = Enumerable.Range(1, 5);
// Result: 1, 2, 3, 4, 5
```

Set Operators

Tuesday, February 6, 2024 2:21 PM

The following operators belong to Set operators category
Distinct
Union
Intersect
Except

Distinct :

```
int[] numbers = { 1, 2, 2, 3, 4, 4, 5 };
var uniqueNumbers = numbers.Distinct();

// Result: 1, 2, 3, 4, 5
```

Union

The Union operator combines two sequences into one sequence that contains the unique elements from both sequences. It essentially performs a set union operation.

```
int[] first = { 1, 2, 3 };
int[] second = { 3, 4, 5 };
var combinedUnique = first.Union(second);

// Result: 1, 2, 3, 4, 5
```

Intersect

The Intersect operator returns a new sequence that contains the elements that appear in both of the original sequences. It is used to find the common elements between two sequences.

```
int[] first = { 1, 2, 3, 4 };
int[] second = { 3, 4, 5, 6 };
var commonElements = first.Intersect(second);

// Result: 3, 4
```

Except

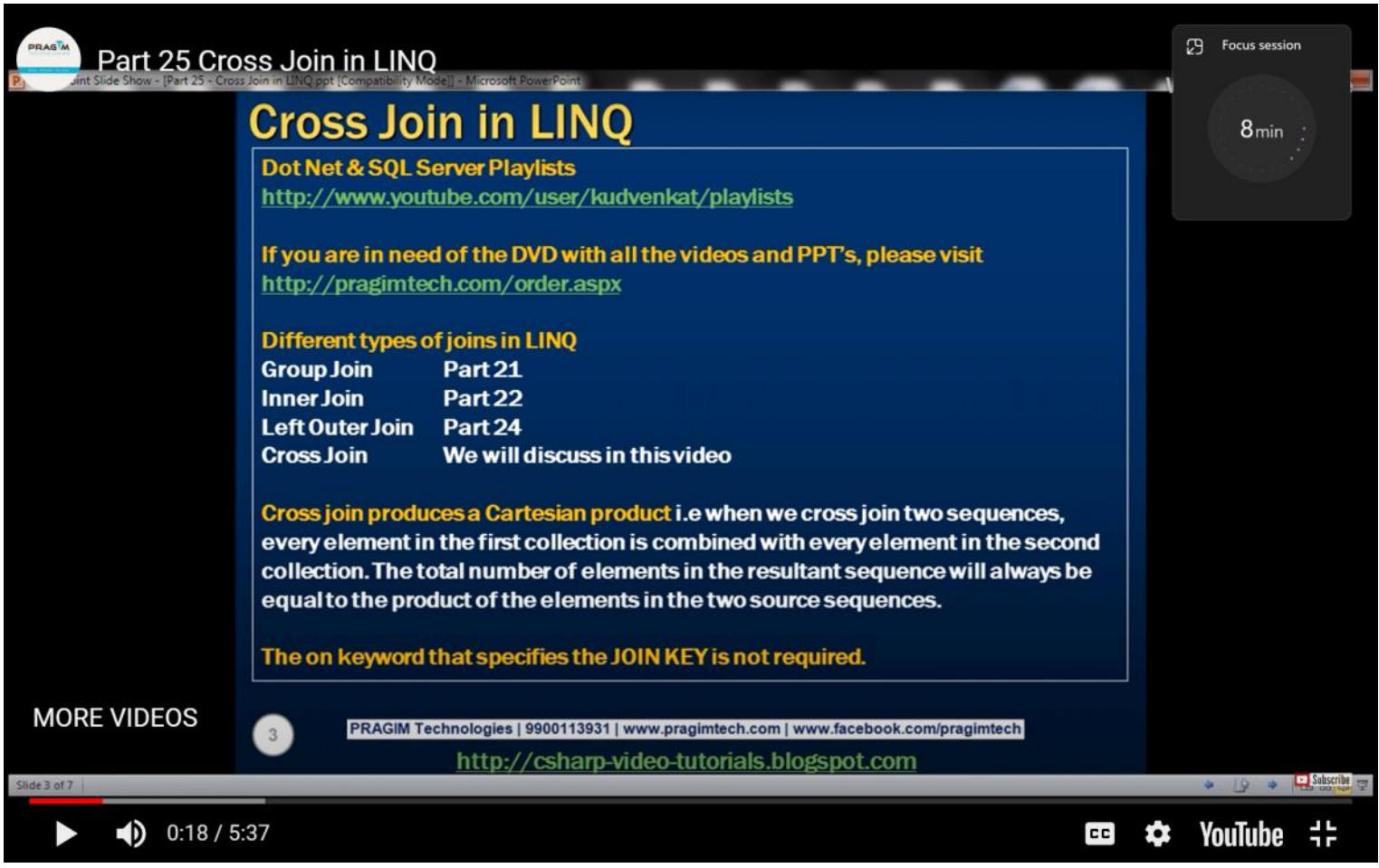
The Except operator returns a new sequence containing the elements from the first sequence that do not appear in the second sequence. It essentially performs a set difference operation.

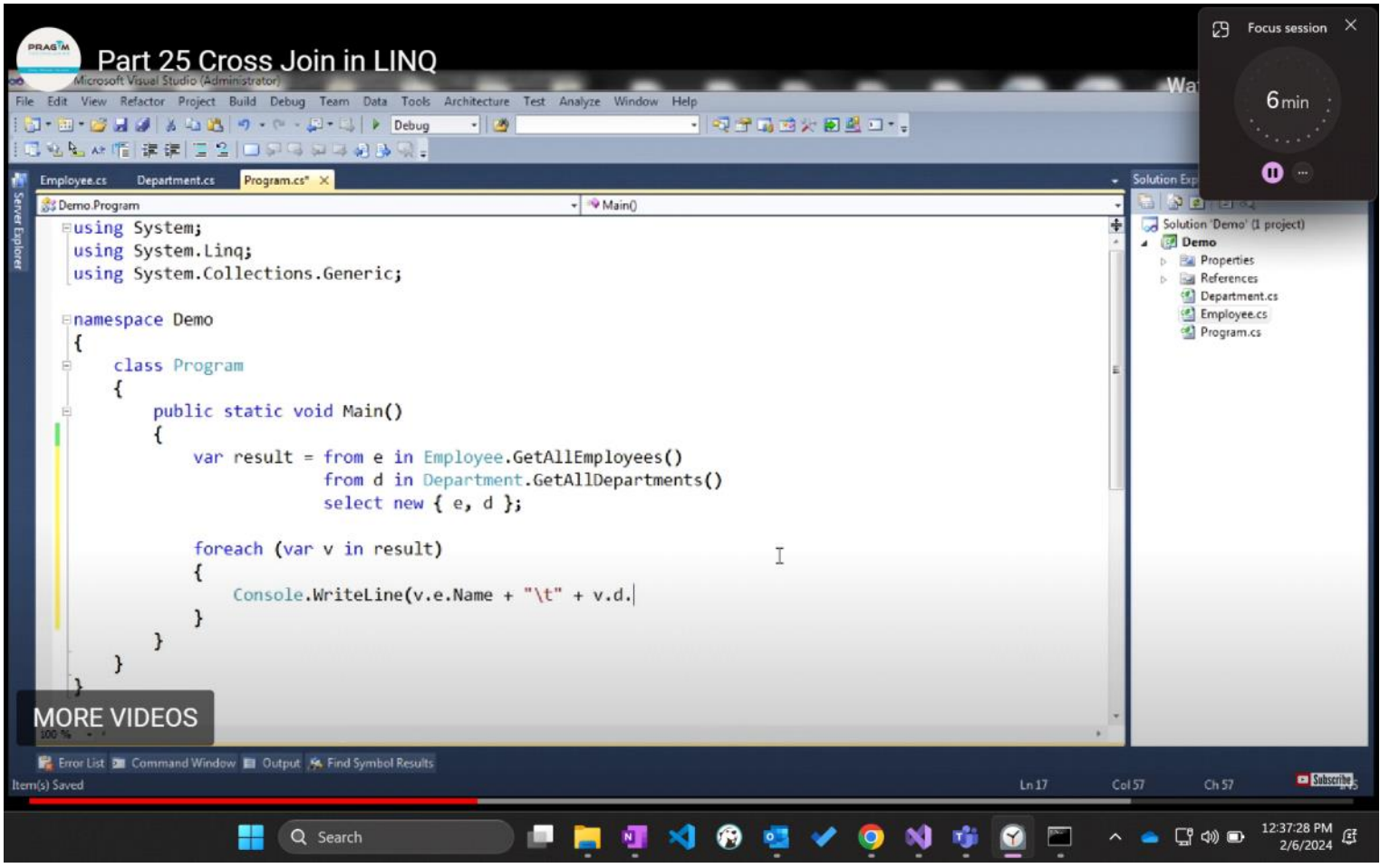
```
int[] first = { 1, 2, 3, 4 };
```

```
int[] second = { 3, 4, 5, 6 };
var uniqueToFirst = first.Except(second);

// Result: 1, 2
```

CROSS JOIN

Tuesday, February 6, 2024 12:34 PM

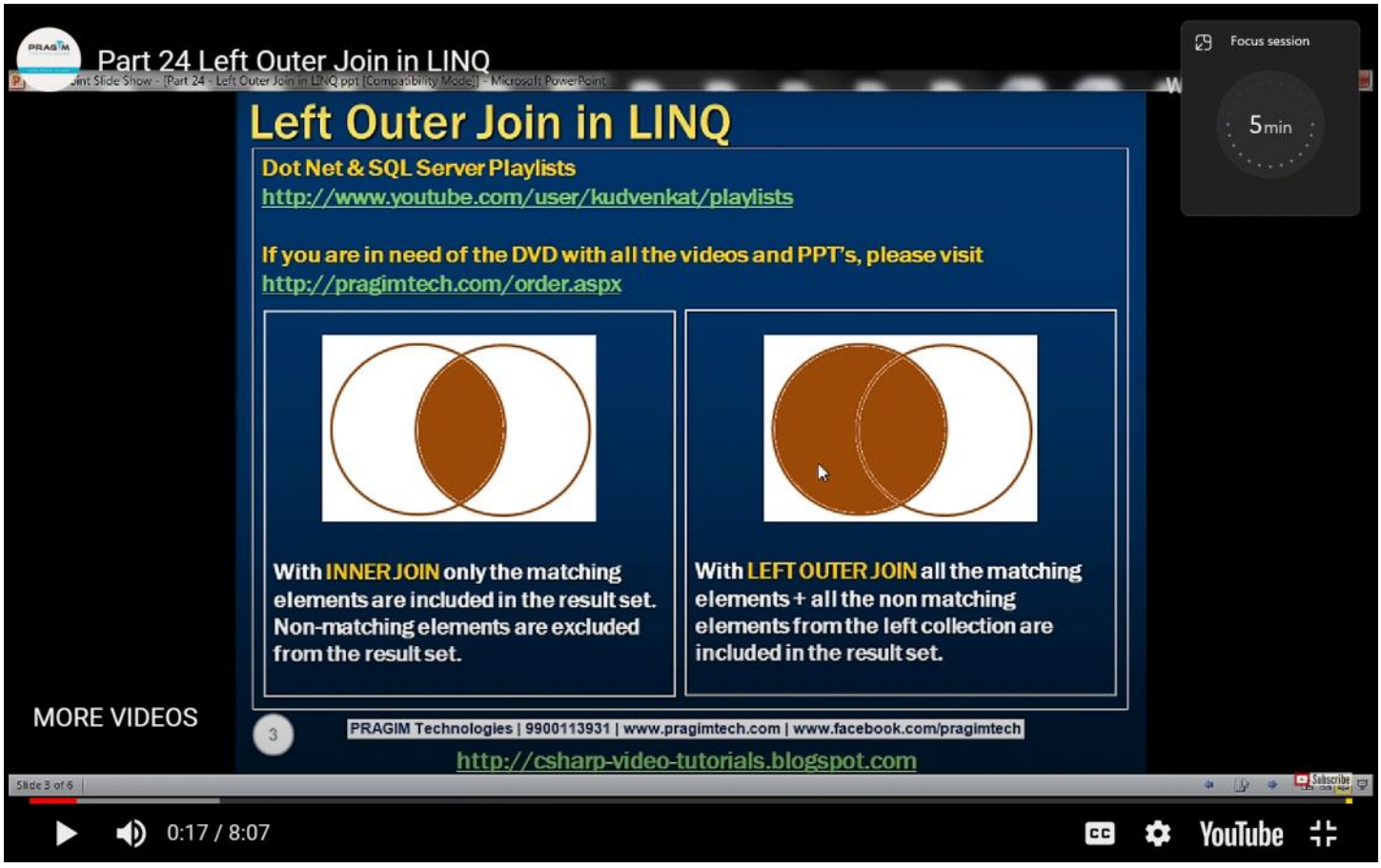




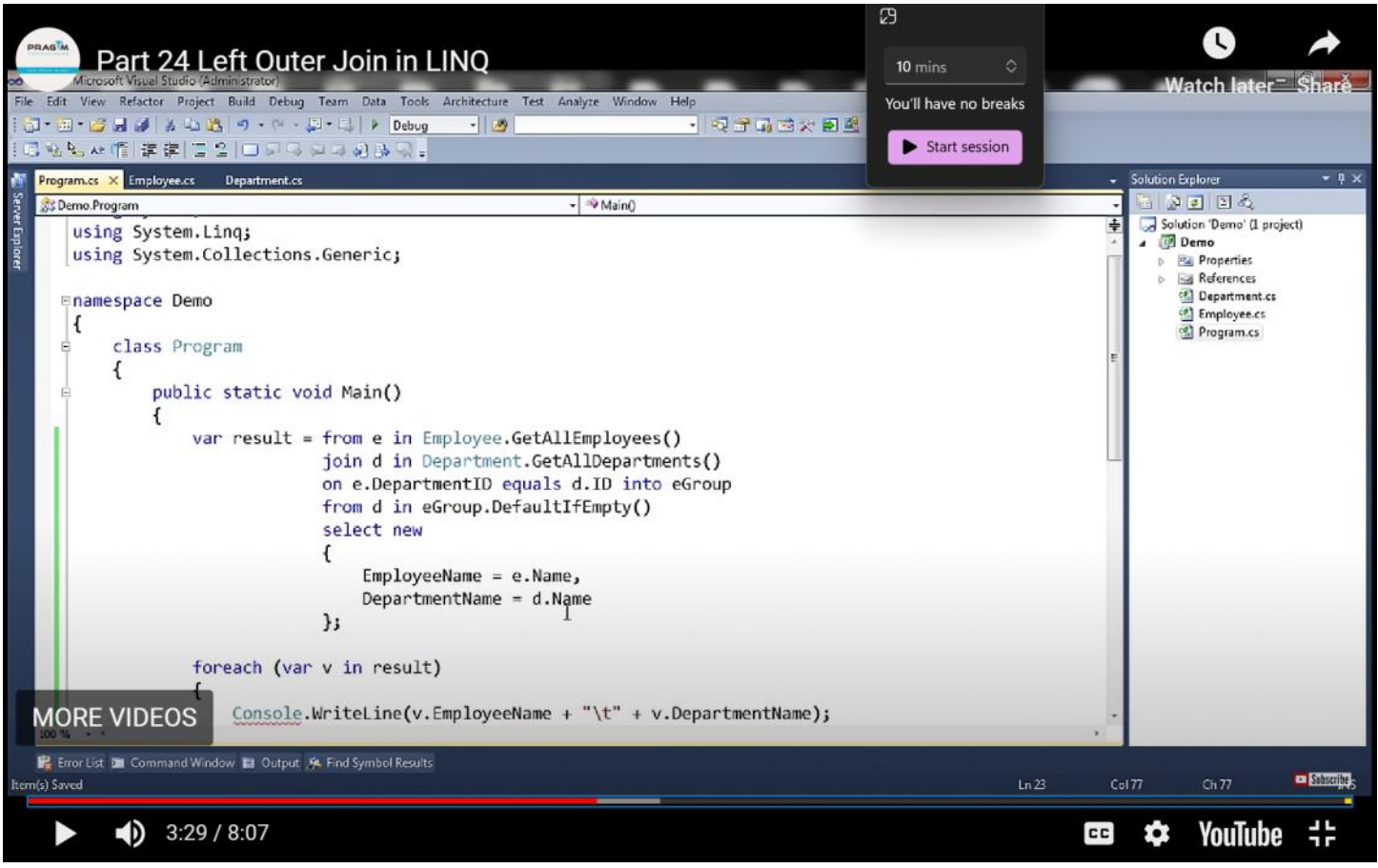

```
var result = from a in authors
        from b in books
        select new
        {
          a,
          b
        };
foreach (var v in result)
{
   Console.WriteLine(v.a.Name +" "+v.b.Title);
}
```

LEFT OUTER JOIN

Tuesday, February 6, 2024 12:28 PM

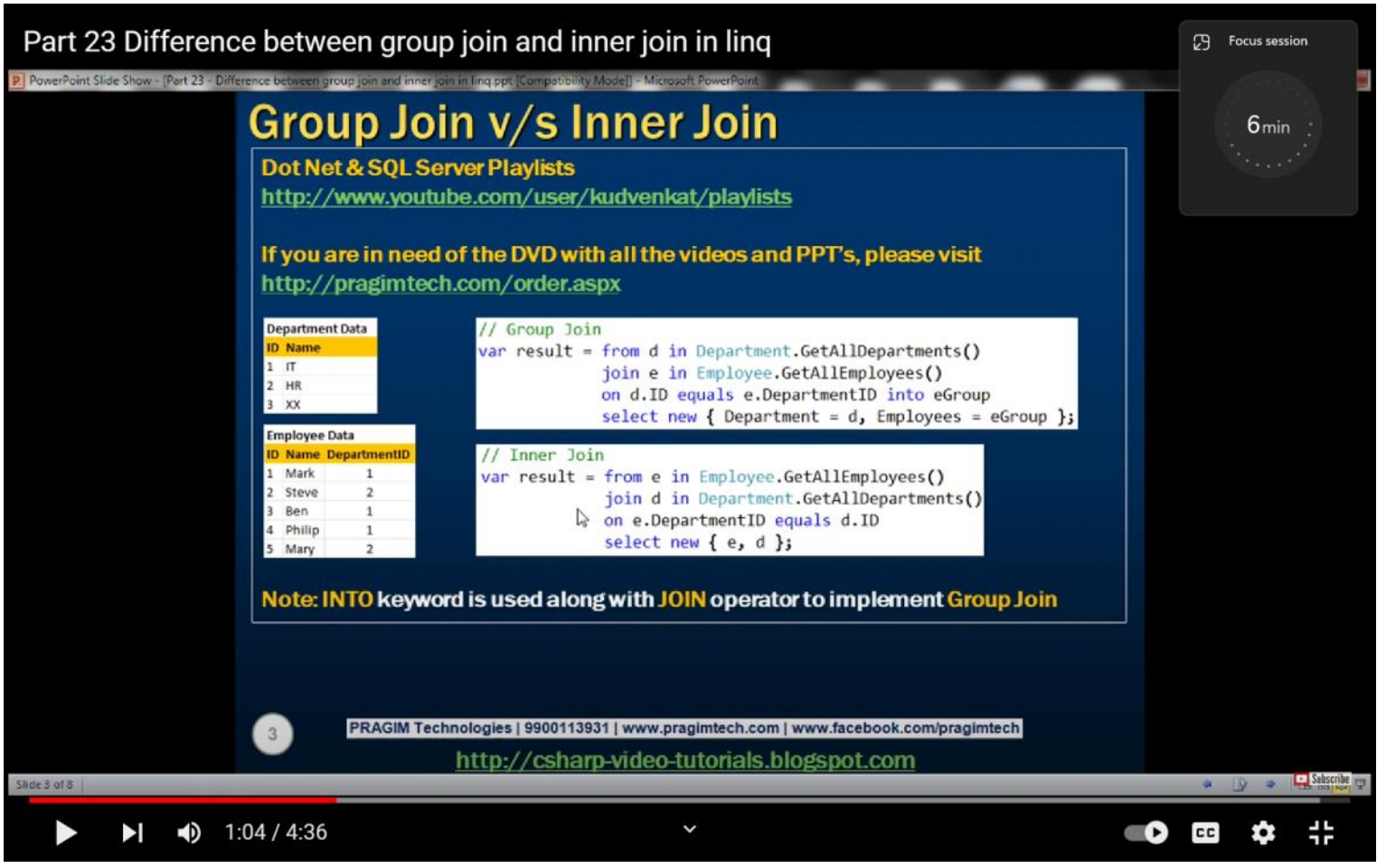


A Left Outer Join in LINQ is used to combine two sequences (collections) based on a key and includes all elements from the first sequence. If there are no matching elements in the second sequence, the result will still include elements from the first sequence, but the corresponding elements from the second sequence will be null or a default value.


Part 24 Left Outer Join in LINQ
Microsoft Visual Studio (Administrator)
10 mins
You'll have no breaks
Start session
Watch later
Share
Program.cs
Employee.cs
Department.cs
Demo.Program
Main()
Solution Explorer
Solution 'Demo' (1 project)
Demo
Properties
References
Department.cs
Employee.cs
Program.cs
using System.Linq;
using System.Collections.Generic;
namespace Demo
{
class Program
{
public static void Main()
{
var result = from e in Employee.GetAllEmployees()
join d in Department.GetAllDepartments()
on e.DepartmentID equals d.ID into eGroup
from d in eGroup.DefaultIfEmpty()
select new
{
EmployeeName = e.Name,
DepartmentName = d.Name
};
foreach (var v in result)
{
Console.WriteLine(v.EmployeeName + "\t" + v.DepartmentName);
MORE VIDEOS
Error List
Command Window
Output
Find Symbol Results
Item(s) Saved
Ln 23
Col 77
Ch 77
3:29 / 8:07
YouTube

Group Join vs Inner Join

Tuesday, February 6, 2024 12:25 PM



Inner Join

Tuesday, February 6, 2024 11:39 AM

Imagine you have two boxes of puzzle pieces. One box has pieces that are parts of animals, and the other box has pieces that are parts of their homes. Now, we want to put together pieces that match: a piece from the animal box that fits with a piece from the home box, so we can see which animal lives in which home.
An "Inner Join" in LINQ is like looking for matching puzzle pieces between these two boxes. We only care about the pieces that fit together perfectly. If a piece from the animal box doesn't have a matching piece in the home box, we set it aside and don't use it. Similarly, if there's a home piece without a matching animal piece, we also set it aside.
So, if we find a piece with a picture of a dog in the animal box, and a piece with a picture of a dog house in the home box, we put them together because they match. But if we have a piece with a picture of a fish, and there's no fishbowl piece in the home box, we can't use the fish piece for our puzzle.
Using an "Inner Join" helps us see only the complete pictures (matches) of animals with their homes, making sure every piece we look at is part of a full set.

```
var innerJoinQuery = from book in books
                     join author in authors on book.AuthorId equals author.AuthorId
                     select new { BookTitle = book.Title, AuthorName = author.Name };
```

GROUP JOIN

Tuesday, February 6, 2024 11:27 AM

Imagine you're having a big party and you want to invite all your friends and their favorite toys. But, each friend can bring more than one toy, and some friends might not bring any toys at all.
A "Group Join" is like making a list for your party that shows each friend and all the toys they're bringing with them in a group. If a friend is coming without any toys, they still get a spot on your list, but their spot for toys just stays empty.
So, if you have a friend named Alex who is bringing a toy car and a doll, and another friend named Sam who isn't bringing any toys, your party list would look like this:

- **Alex: Toy Car, Doll**
- **Sam:**

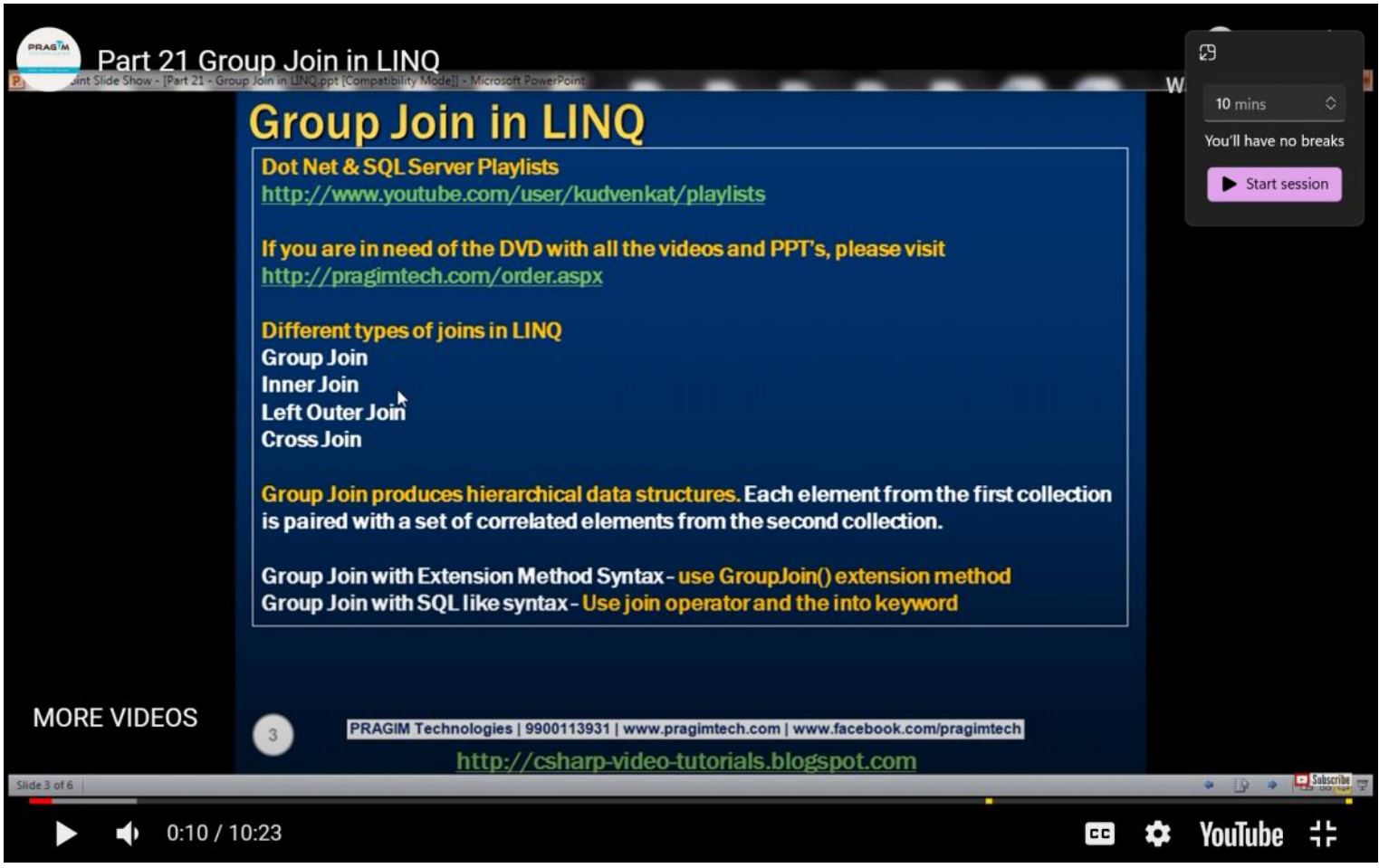
This way, you can see all your friends and what toys they're bringing to the party all at once. And even if some friends don't bring toys, you still know they're coming to your party. That's what a "Group Join" does when we're organizing our information!

```
public class Author
{
   public int AuthorId { get; set; }
   public string Name { get; set; }
}

public class Book
{
   public string Title { get; set; }
   public int AuthorId { get; set; }
}

List<Author> authors = new List<Author>
{
   new Author { AuthorId = 1, Name = "Author One" },
   new Author { AuthorId = 2, Name = "Author Two" },
   new Author { AuthorId = 3, Name = "Author Three" }
};

List<Book> books = new List<Book>
{
   new Book { Title = "Book A", AuthorId = 1 },
   new Book { Title = "Book B", AuthorId = 1 },
   new Book { Title = "Book C", AuthorId = 2 }
};
```

```
var groupJoinQuery = from author in authors
                     join book in books on author.AuthorId equals book.AuthorId into bookGroup
                     select new
                     {
                        AuthorName = author.Name,
                        Books = bookGroup
                     };

foreach (var authorGroup in groupJoinQuery)
{
   Console.WriteLine($"Author: {authorGroup.AuthorName}");
   foreach (var book in authorGroup.Books)
   {
      Console.WriteLine($"\tBook: {book.Title}");
   }
}
```

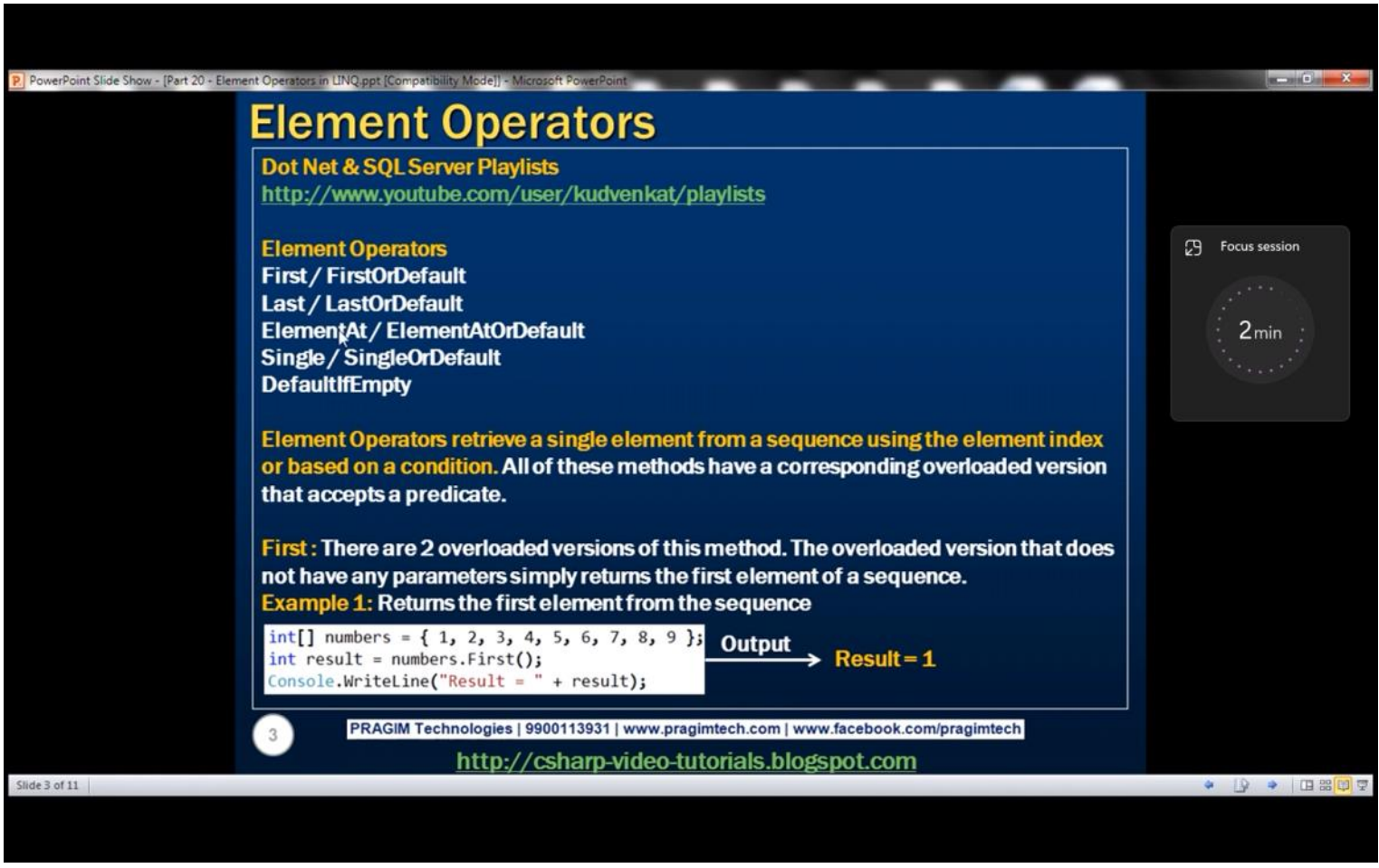
Joins

Tuesday, February 6, 2024 11:27 AM



ELEMENT OPERATORS

Tuesday, February 6, 2024 10:37 AM



Element operators in LINQ are special methods used to select elements from a sequence (like a list or array). These operators make it easy to pick out single items from a collection based on their position or a condition. Here are some of the most commonly used element operators:

1. **First: This operator returns the first element of a sequence. If you provide a condition, it returns the first element that meets that condition. It throws an exception if no element is found.**
2. **FirstOrDefault: Similar to First, but instead of throwing an exception if no element is found, it returns a default value (like null for reference types or 0 for numeric types).**
3. **Last: This operator returns the last element of a sequence. If you provide a condition, it returns the last element that meets that condition. Like First, it throws an exception if no element is found.**
4. **LastOrDefault: Similar to Last, but returns a default value instead of throwing an exception if no element is found.**
5. **Single: This operator is used when you expect exactly one element in the sequence, or exactly one element that meets a condition. It throws an exception if the sequence is empty or if there's more than one element.**
6. **SingleOrDefault: Similar to Single, but returns a default value instead of throwing an exception if the sequence is empty. It still throws an exception if there's more than one element.**
7. **ElementAt: Returns the element at a specified index in the sequence. It throws an exception if the index is out of range.**
8. **ElementAtOrDefault: Similar to ElementAt, but returns a default value instead of throwing an exception if the index is out of range.**
9. **DefaultIF:**
   The DefaultIfEmpty method in LINQ is used to provide a default value for a sequence that contains no elements. When you apply DefaultIfEmpty to an empty sequence, it returns a sequence with a single element: the default value for the type of the sequence. If you do not specify a default value, it uses the default value for the type of the elements in the sequence

```
public static void Main(string[] args)
{
   int[] numbers = { 2, 4, 6, 8, 10 };
   var numberslist = numbers.DefaultIfEmpty(0);
   foreach (int number in numberslist)
   {
      Console.Write(number+" ");
   }
   Console.WriteLine();
   int firstNumber = numbers.First();
   Console.WriteLine(firstNumber);
   // Result: 2
   int firstOddNumber = numbers.FirstOrDefault(n => n % 2 != 0);
   // Result: 0 (since there are no odd numbers, it returns the default value for int, which is 0)
   Console.WriteLine(firstOddNumber);
   int lastNumber = numbers.Last();
   Console.WriteLine(lastNumber);
   // Result: 10
   int lastOddNumber = numbers.LastOrDefault(n => n % 2 != 0);
   // Result: 0 (since there are no odd numbers, it returns the default value for int)
   Console.WriteLine(lastOddNumber);
   int[] oneNumber = { 9 };
   int singleNumber = oneNumber.Single();
   Console.WriteLine(singleNumber);
   // Result: 9
   int[] noNumbers = { };
   int singleOrDefaultNumber = noNumbers.SingleOrDefault();
   // Result: 0 (since the array is empty, it returns the default value for int)
   Console.WriteLine(singleOrDefaultNumber);
   int thirdNumber = numbers.ElementAt(2);
   Console.WriteLine(thirdNumber);
   // Result: 6
   int tenthNumber = numbers.ElementAtOrDefault(9);
   // Result: 0 (since there is no 10th element, it returns the default value for int)
   Console.WriteLine(tenthNumber);

}
```

GroupBy Multiple Keys

Tuesday, February 6, 2024 10:33 AM

Grouping by multiple keys in LINQ allows you to organize your data based on more than one attribute. For example, if you have a list of people, you might want to group them by both their city and their job. This way, you can see all the people from the same city with the same job grouped together.

```
using System;
using System.Collections.Generic;
using System.Linq;

public class Person
{
   public string Name { get; set; }
   public string City { get; set; }
   public string Job { get; set; }
}

public class Program
{
   public static void Main()
   {
      List<Person> people = new List<Person>
      {
         new Person { Name = "Alice", City = "New York", Job = "Developer" },
         new Person { Name = "Bob", City = "Boston", Job = "Designer" },
         new Person { Name = "Charlie", City = "New York", Job = "Developer" },
         new Person { Name = "Diana", City = "Boston", Job = "Developer" },
         new Person { Name = "Eve", City = "New York", Job = "Designer" }
      };

      var groupedPeople = from person in people
                          group person by new { person.City, person.Job } into cityJobGroup
                          select new
                          {
                             City = cityJobGroup.Key.City,
                             Job = cityJobGroup.Key.Job,
                             Persons = cityJobGroup.ToList()
                          };

      foreach (var group in groupedPeople)
      {
         Console.WriteLine($"City: {group.City}, Job: {group.Job}");
         foreach (var person in group.Persons)
         {
```

```
            Console.WriteLine($" - {person.Name}");
        }
    }
  }
}
```

GroupBY

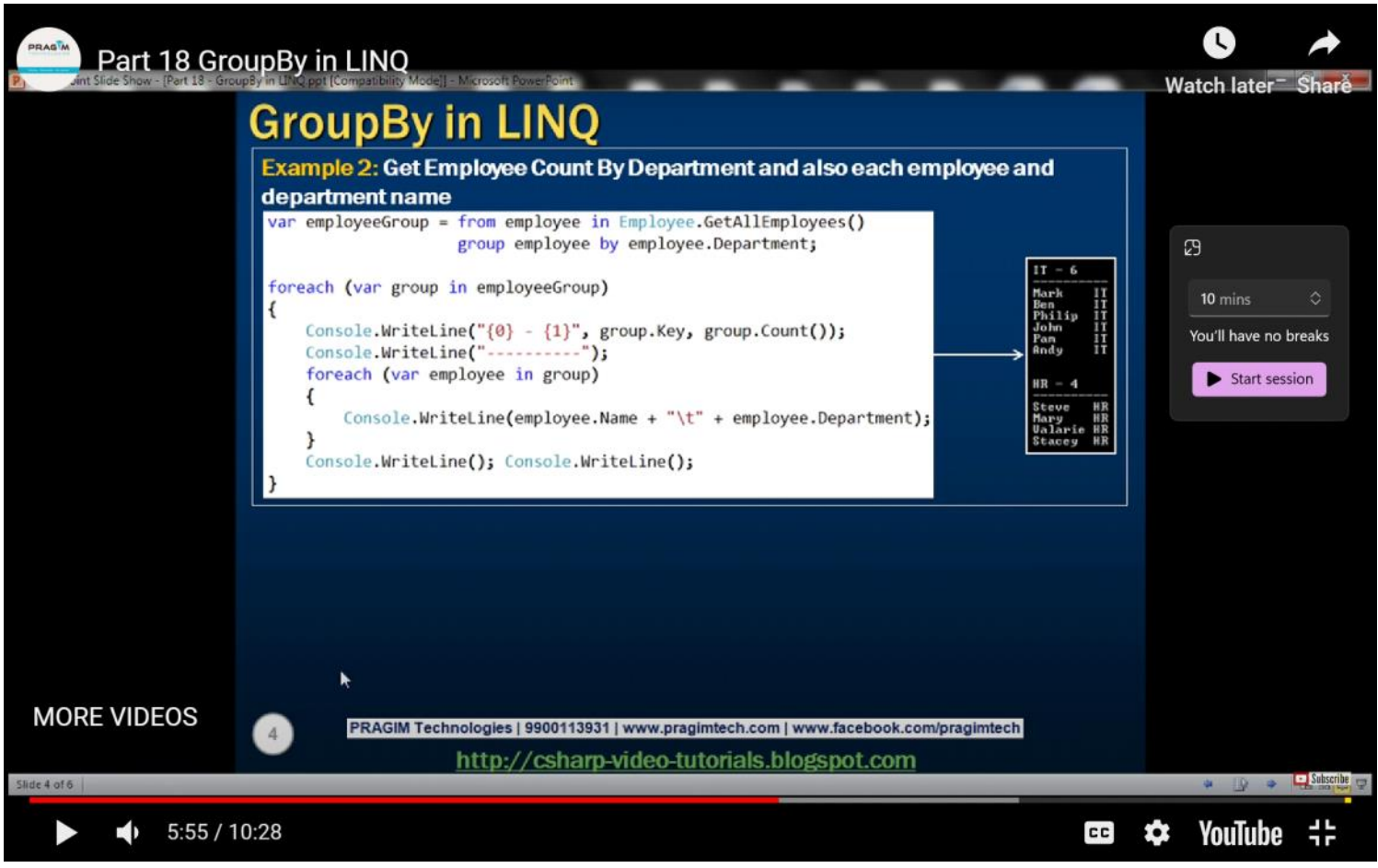
Tuesday, February 6, 2024 9:45 AM

Imagine you have a big box of crayons. There are lots of different colors: red, blue, green, and so on. But right now, all the crayons are mixed up, and you want to organize them so that all the crayons of the same color are together. This will make it easier for you to find the color you want when you're drawing.
In computer programming, we sometimes have a similar problem. We have a lot of data (like our crayons), and we want to organize it based on some common feature (like the color of the crayons). This is where "Group By" comes into play, but instead of crayons, we're working with information.

Grouping data in LINQ (Language Integrated Query) is a powerful feature that allows you to organize your data into categories based on a specified key.

EXAMPLE :

```
List<Person> people = new List<Person>
{
    new Person { Name = "Alice", Age = 30 },
    new Person { Name = "Bob", Age = 25 },
    new Person { Name = "Charlie", Age = 30 },
    new Person { Name = "David", Age = 25 },
    new Person { Name = "Eve", Age = 35 }
};


var groupedByAge = from person in people
                   group person by person.Age;


foreach (var group in groupedByAge)
{
    Console.WriteLine($"Age Group: {group.Key} and count is {group.Count()}");

}
```


Part 18 GroupBy in LINQ
Watch later
Share
GroupBy in LINQ
Example 2: Get Employee Count By Department and also each employee and department name
var employeeGroup = from employee in Employee.GetAllEmployees()
group employee by employee.Department;
foreach (var group in employeeGroup)
{
Console.WriteLine("{0} - {1}", group.Key, group.Count());
Console.WriteLine("----------");
foreach (var employee in group)
{
Console.WriteLine(employee.Name + "\t" + employee.Department);
}
Console.WriteLine(); Console.WriteLine();
}
IT - 6
Mark IT
Ben IT
Philip IT
John IT
Pan IT
Andy IT
HR - 4
Steve HR
Mary HR
Valarie HR
Stacey HR
10 mins
You'll have no breaks
Start session
MORE VIDEOS
4
PRAGIM Technologies | 9900113931 | www.pragimtech.com | www.facebook.com/pragimtech
http://csharp-video-tutorials.blogspot.com
Slide 4 of 6
5:55 / 10:28
YouTube

ASEnumerable vs ASQuarable

Tuesday, February 6, 2024 9:54 AM

Imagine you have a big box of toy blocks. Some blocks are inside the box (your home), and some blocks are at your friend's house (a database far away).

- **When you use AsEnumerable, it's like bringing all the toy blocks from your friend's house to your home before you start building something. You bring everything over, even if you're going to use just a few blocks to build a small tower. This means you might end up moving a lot of blocks that you don't really need, which can take a lot of time and effort.**
- **On the other hand, using AsQueryable is like calling your friend and asking them to select just the blocks you need and then bringing those few blocks to your house. This way, you only get the blocks you're going to use, which saves a lot of time and effort because you're not moving unnecessary blocks.**

OfType

Monday, February 5, 2024 6:37 PM

OfType is also same as Cast .

Cast is throwiing the errors when it sees another / other types of data types .

But OfTypes only sees that types and give outputs

```
class Program
{

   public static void Main(string[] args)
   {
      ArrayList list = new ArrayList();
      list.Add(1);
      list.Add(2);
      list.Add(3);
      list.Add(4);
      list.Add("A");
      list.Add("B");

      var result=list.OfType<int>();
      foreach(int i in result)
      {
         Console.WriteLine(i);
      }
   }
```

Output :

```
1
2
3
4
```

It ignores other types of data types .

Conversion Operators

Monday, February 5, 2024 5:38 PM

The most commonly used conversion operators include ToArray, ToList, ToDictionary, and Cast.

ToArray

- **Purpose: Converts the result of a LINQ query to an array.**
- **Use Case: When you need a fixed-size collection or when you're working with APIs that require arrays.**

```
var numbers = new List<int> { 1, 2, 3, 4, 5 };
var numbersArray = numbers.ToArray();
```

ToList

- **Purpose: Converts the result of a LINQ query to a List<T>.**
- **Use Case: When you need to work with a collection that supports indexing or when you need a collection that can change size.**

```
var numbers = new List<int> { 1, 2, 3, 4, 5 };
var numbersList = numbers.Where(n => n > 2).ToList();
```

ToDictionary

- **Purpose: Converts the result of a LINQ query to a Dictionary<TKey, TValue>.**
- **Use Case: When you need to access elements by a specific key quickly.**

```
var people = new List<Person>
{
   new Person { Name = "John", Age = 30 },
   new Person { Name = "Jane", Age = 25 }
};
var dictionary = people.ToDictionary(p => p.Name, p => p.Age);
```

Cast

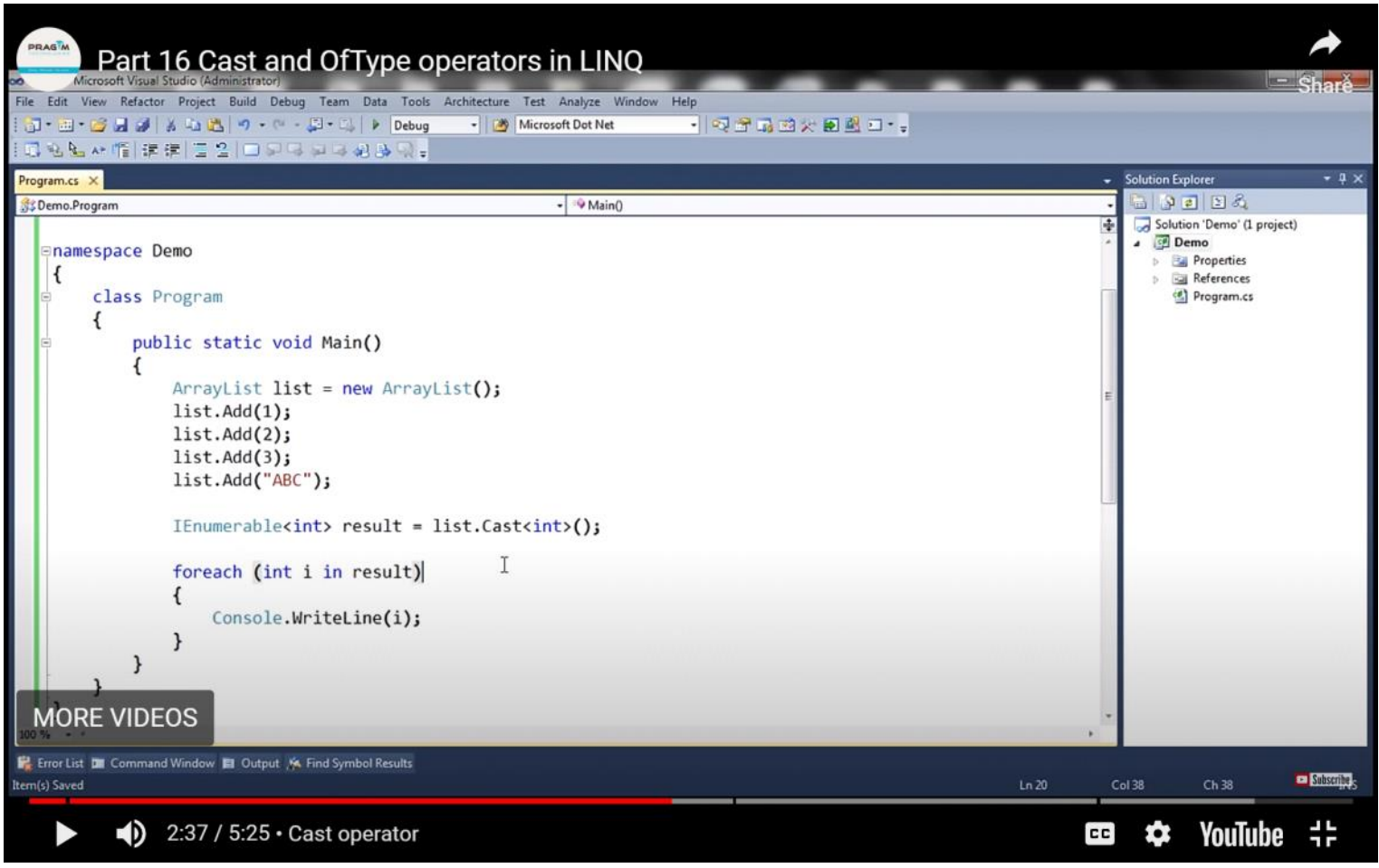
- **Purpose: Converts the elements of an IEnumerable to the specified type.**
- **Use Case: When you have a non-generic collection and you need to work with the elements in a strongly-typed manner.**

```
IEnumerable objects = new List<object> { 1, 2, 3, 4, 5 };
var numbers = objects.Cast<int>();
```

CAST

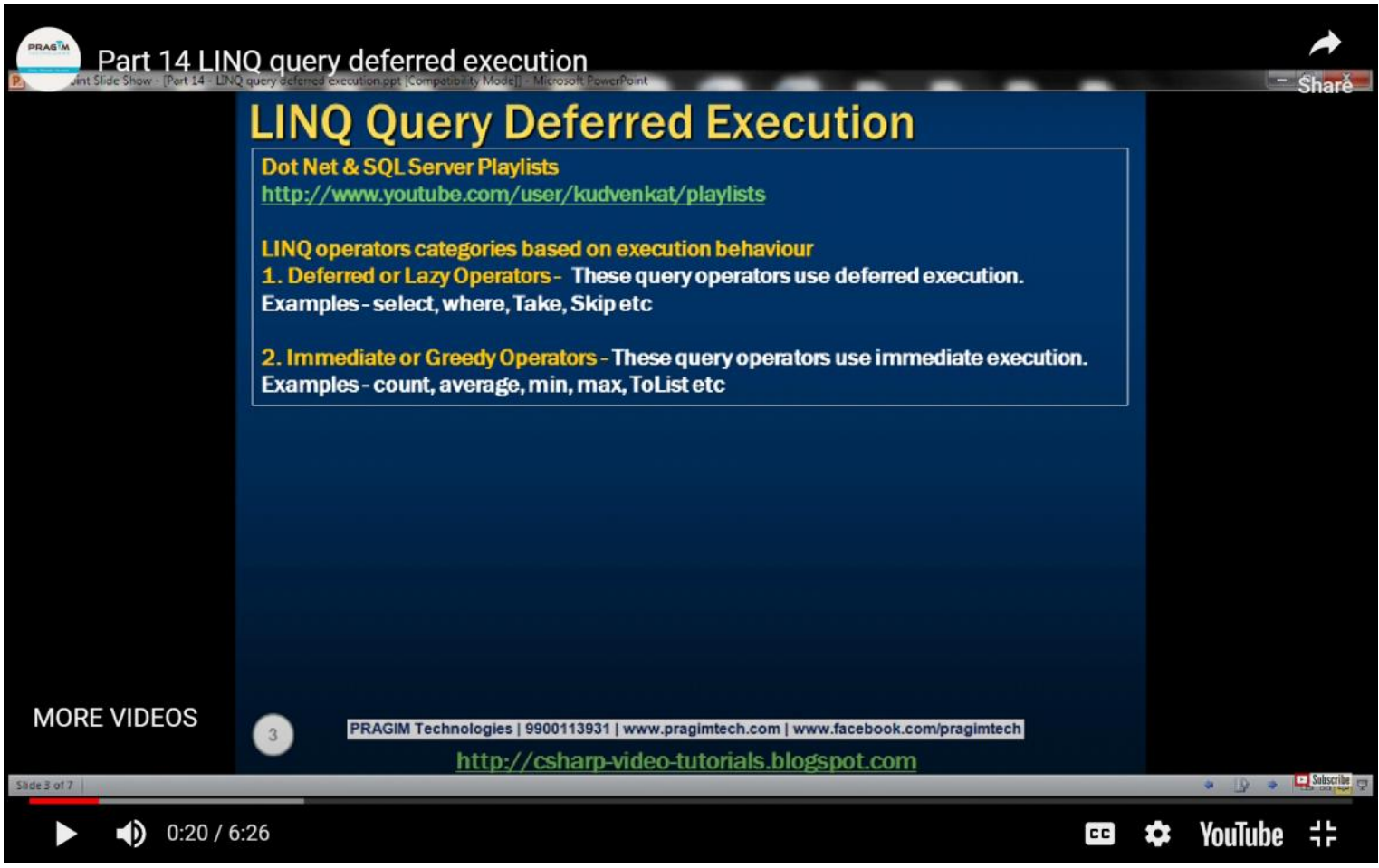
Monday, February 5, 2024 5:37 PM

The "Cast" is like magic glasses that you put on, and suddenly, you can see which wrapped toys are cars. When you look through these glasses, all the toys that are not cars become invisible, and only the cars remain visible to you. So, you can easily pick out all the cars and play with them.
In computer terms, when we have a box of mixed things (like numbers, letters, or any items), and we only want items of a certain type (like just the numbers), we use "Cast" to see only those items. It's like telling the computer, "Please show me only the items that are numbers and ignore everything else."



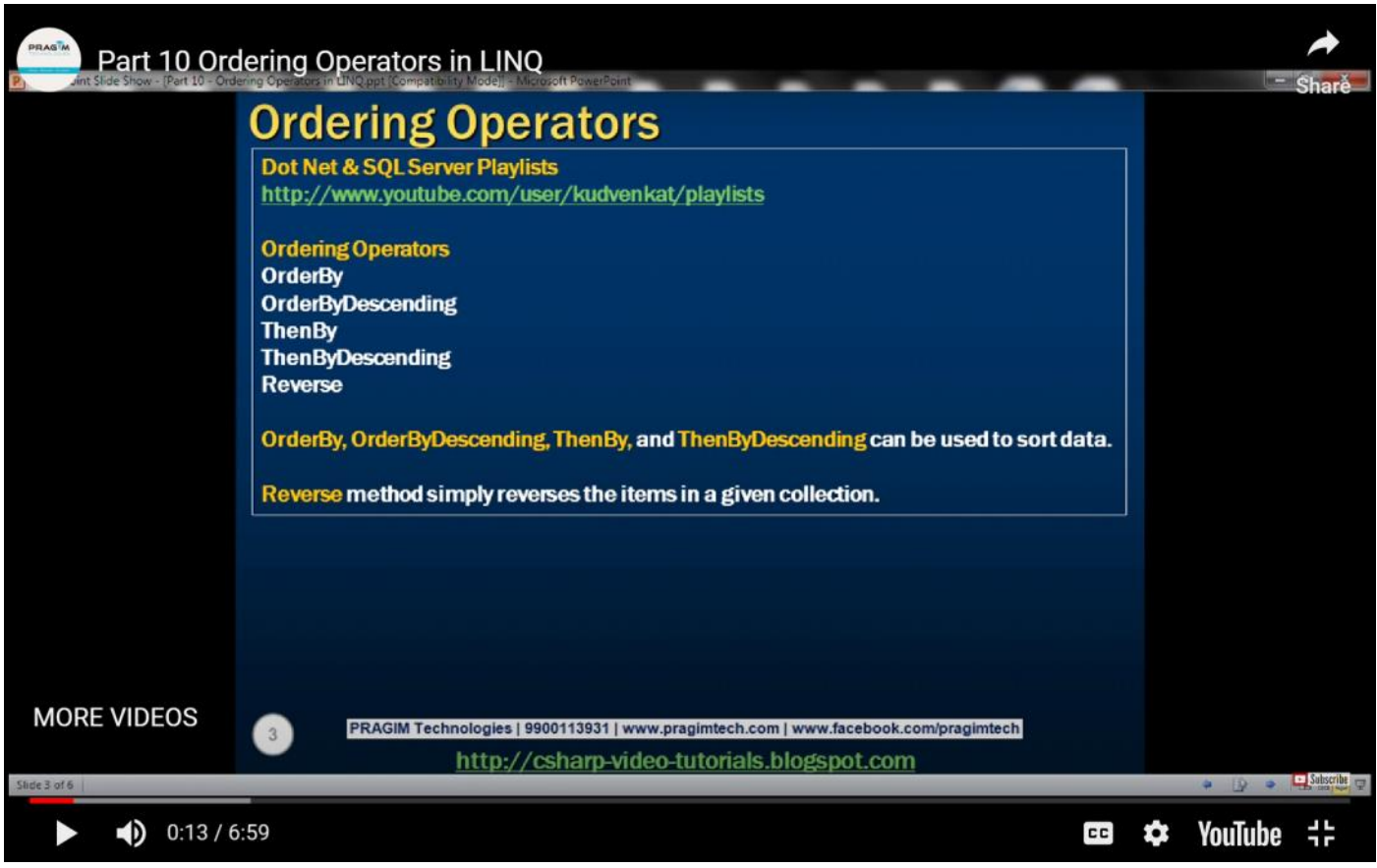
Deferred Operators

Monday, February 5, 2024 4:06 PM



Ordering Operators In List

Monday, February 5, 2024 3:09 PM



OrderBy :

- **Purpose: Sorts the elements of a sequence in ascending order according to a key.**
- **Example: Sorting a list of numbers from lowest to highest or sorting a list of names alphabetically.**

```
var numbers = new List<int> { 3, 1, 4, 1, 5, 9, 2 };
var sortedNumbers = numbers.OrderBy(n => n);

var people = new List<string> { "Alice", "Bob", "Charlie" };
var sortedPeople = people.OrderBy(name => name);
```

OrderByDescending

- **Purpose: Sorts the elements of a sequence in descending order according to a key.**
- **Example: Sorting a list of numbers from highest to lowest.**

```
var sortedNumbersDescending = numbers.OrderByDescending(n => n);
```

ThenBy

- **Purpose: Performs a secondary sort in ascending order on a sequence that has already been sorted.**
- **Use Case: Useful when you want to sort by one property first and then by another property within each group of the first sort key.**

```
var people = new List<Person>
{
   new Person { FirstName = "John", LastName = "Doe" },
   new Person { FirstName = "Jane", LastName = "Doe" },
   new Person { FirstName = "John", LastName = "Smith" }
};
var sortedPeople = people.OrderBy(p => p.LastName).ThenBy(p => p.FirstName);
```

ThenByDescending

- **Purpose: Performs a secondary sort in descending order on a sequence that has already been sorted.**
- **Use Case: Similar to ThenBy, but sorts the second key in descending order.**

```
var sortedPeopleDescending = people.OrderBy(p => p.LastName).ThenByDescending(p => p.FirstName);`
```

- **ThenBy and ThenByDescending are used for subsequent sorting operations when you need to sort by additional keys. These are applied to sequences that have already been sorted by a previous OrderBy or OrderByDescending.**
- **LINQ's ordering operators work by internally generating and using IComparer implementations that compare the keys extracted from the elements of the sequence.**

Reverse :

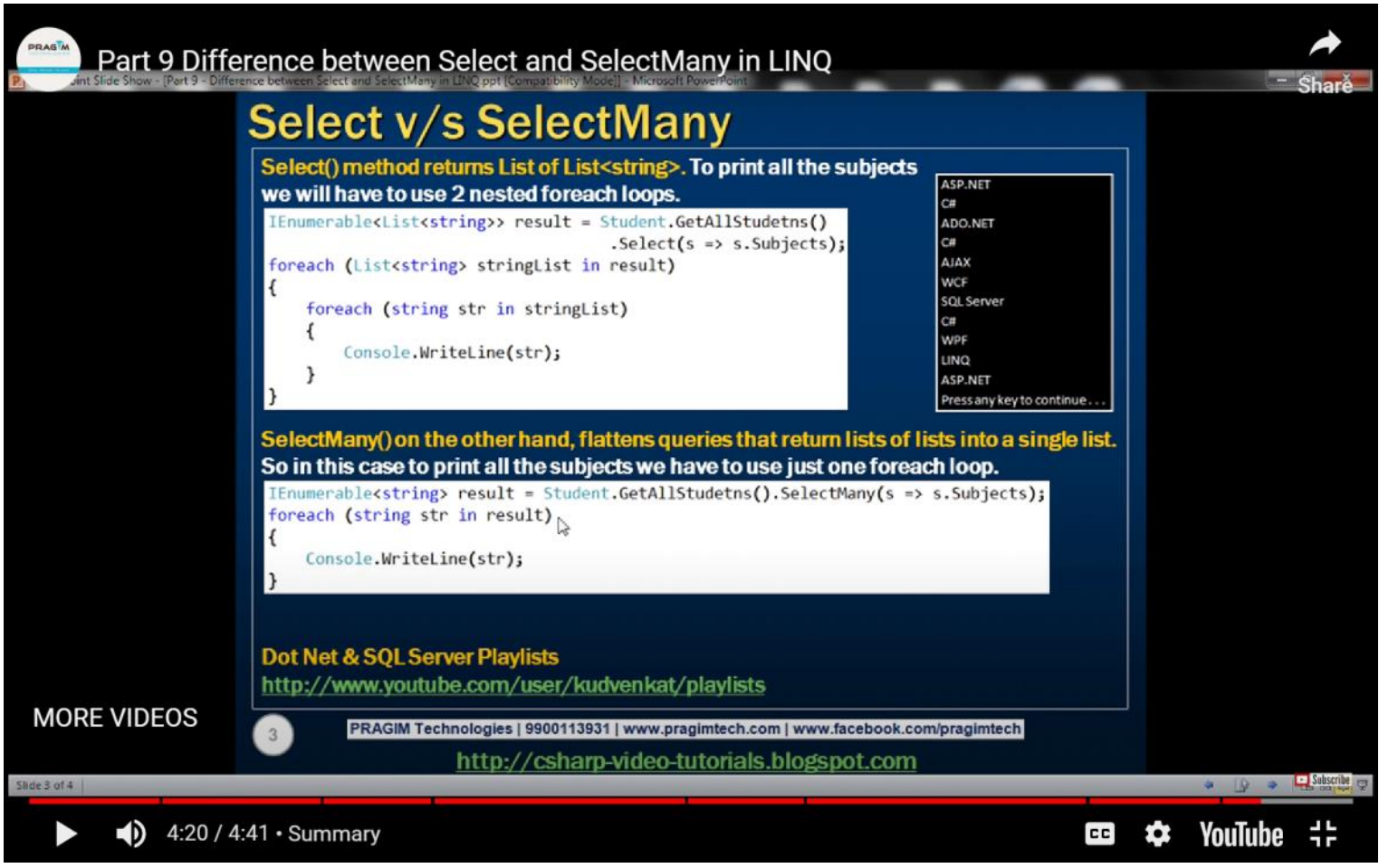
The Reverse method in LINQ (Language Integrated Query) is used to invert the order of elements in a collection

From <https://chat.openai.com/c/ffbbbef2-8896-4388-a3da-44a444fbdc46>

```
using System;
using System.Linq;

class Program
{
   static void Main()
   {
      int[] numbers = { 1, 2, 3, 4, 5 };

      // Reverse the array
      var reversedNumbers = numbers.Reverse();

      foreach (var number in reversedNumbers)
      {
         Console.WriteLine(number);
      }
      // Output will be: 5, 4, 3, 2, 1
   }
}
```

Select vs Select Many

Monday, February 5, 2024 3:08 PM



Projection Operators in LinQ

Monday, February 5, 2024 2:25 PM

Projection operators in LINQ (Language Integrated Query) are used to transform elements in a collection into a new form. Essentially, these operators allow you to shape the data into a new format or extract specific information from each element in the collection. The most commonly used projection operators in LINQ are Select and SelectMany.

Select

The Select operator is used to project each element of a sequence into a new form. This is similar to the SELECT statement in SQL, but it can be used more flexibly to transform data into new shapes, such as selecting specific properties from objects, transforming types, or performing calculations on elements.

```
using System;
using System.Linq;

class Program
{
   static void Main()
   {
      string[] names = { "John", "Jane", "Jake" };

      // Project each name into a new form with a greeting
      var greetings = names.Select(name => $"Hello, {name}!");

      foreach (var greeting in greetings)
      {
         Console.WriteLine(greeting);
      }
   }
}
```

SELECT MANY :

The SelectMany operator is used to project each element of a sequence to an IEnumerable<T> and then flattens the resulting sequences into one sequence. This is useful when working with collections of collections, allowing you to perform a projection on each element and then flatten the result into a single collection.

```
using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
   static void Main()
   {
      List<List<int>> numbersList = new List<List<int>>
```

```
        {
            new List<int> { 1, 2, 3 },
            new List<int> { 4, 5, 6 },
            new List<int> { 7, 8, 9 }
        };

        // Flatten the list of lists into a single sequence of numbers
        var allNumbers = numbersList.SelectMany(subList => subList);

        foreach (var number in allNumbers)
        {
            Console.WriteLine(number);
        }
    }
}


using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
    static void Main()
    {
        List<Book> books = new List<Book>
        {
            new Book { Title = "C# Programming", Authors = new List<string> { "Author A", "Author B" } },
            new Book { Title = "LINQ Magic", Authors = new List<string> { "Author C", "Author A" } },
            new Book { Title = "ASP.NET Core", Authors = new List<string> { "Author B", "Author D" } }
        };

        // Project each book into a new form
        var bookSummaries = books.Select(b => new { b.Title, AuthorCount = b.Authors.Count });

        foreach (var summary in bookSummaries)
        {
            Console.WriteLine($"{summary.Title} has {summary.AuthorCount} authors.");
        }

        // Create a list of unique authors from all books
        var allAuthors = books.SelectMany(b => b.Authors).Distinct();

        Console.WriteLine("\nAuthors:");
        foreach (var author in allAuthors)
        {
            Console.WriteLine(author);
```

Restriction Operators

Monday, February 5, 2024 12:34 PM

Restriction Operators are like a magic tool that helps find only the toys (or in grown-up terms, "data") you want to play with from a huge box (or "database"). They can pick out toys based on what you tell them. For example, if you say you only want red toys, the Restriction Operator will search through the box and show you all the red toys. If you say you want toys that are not broken, it will find those for you.

Where: This operator is used to filter a sequence (such as a list, array, or any other enumerable collection) based on a predicate. A predicate is a function that evaluates each element in the sequence to true or false, determining whether the element should be included in the result. The Where method iterates over each element in the collection, applying the predicate to each one. Elements that make the predicate true are included in the resulting collection.

Example :

```
using System;
using System.Linq;

class Program
{
   static void Main()
   {
      int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

      // Use the Where operator to find even numbers
      var evenNumbers = numbers.Where(n => n % 2 == 0);

      Console.WriteLine("Even numbers:");
      foreach (var num in evenNumbers)
      {
         Console.WriteLine(num);
      }
   }
}
```

Aggregate Function

Monday, February 5, 2024 12:19 PM

Like Foreach loop …..

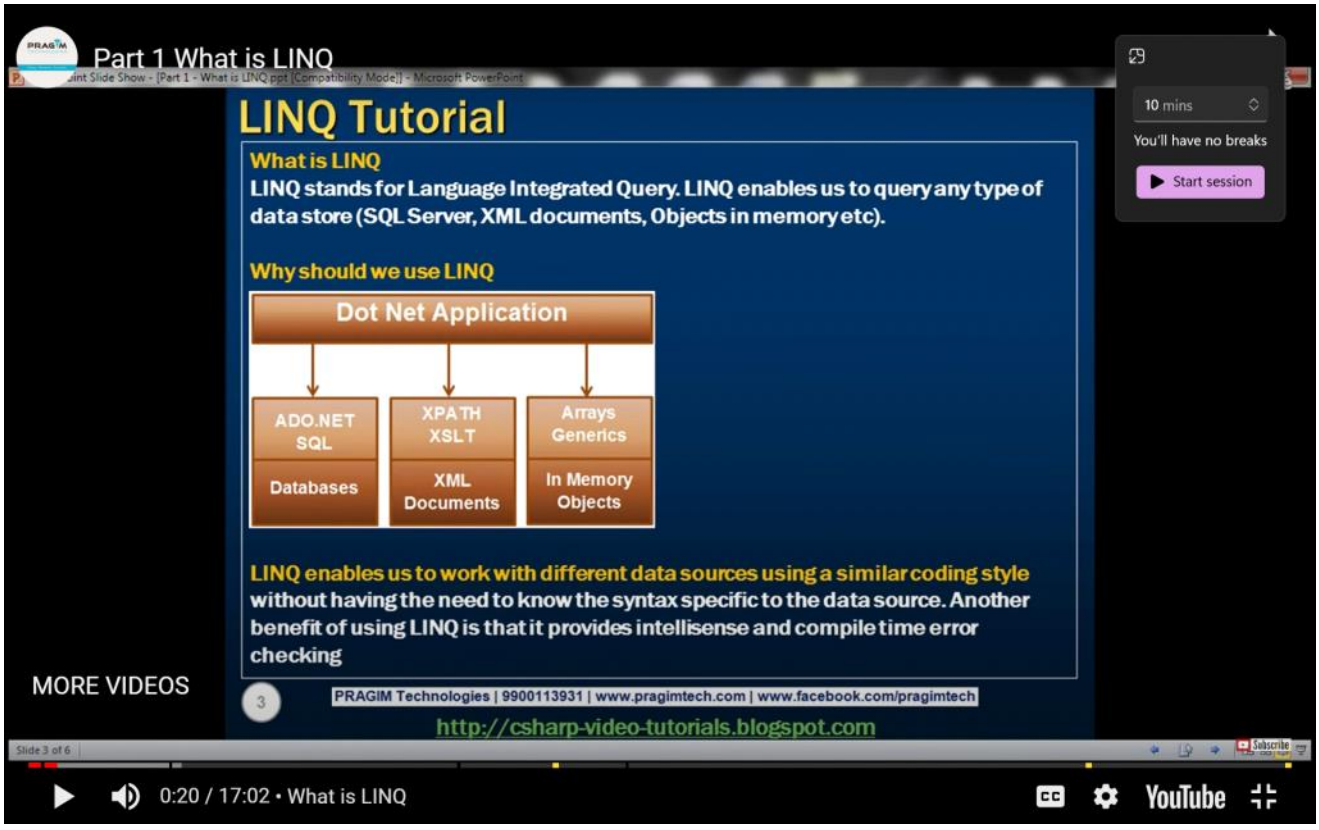
It is going to do that until it reaches the last element .

```
string result = countries.Aggregate((a,b)=> a +" "+b);
Console.WriteLine(result);
```

```
int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, };

int  result1=numbers.Aggregate((a,b)=>a*b);
int result4 = numbers.Aggregate(10,(a,b)=>a+b);
int result2 = numbers.Aggregate((a, b) => a + b);
int result3 = numbers.Aggregate((a, b) => a % b);
Console.WriteLine(result1);
Console.WriteLine(result2);
Console.WriteLine(result3);
Console.WriteLine(result4);
```

What is LinQ?

Friday, February 2, 2024 4:55 PM

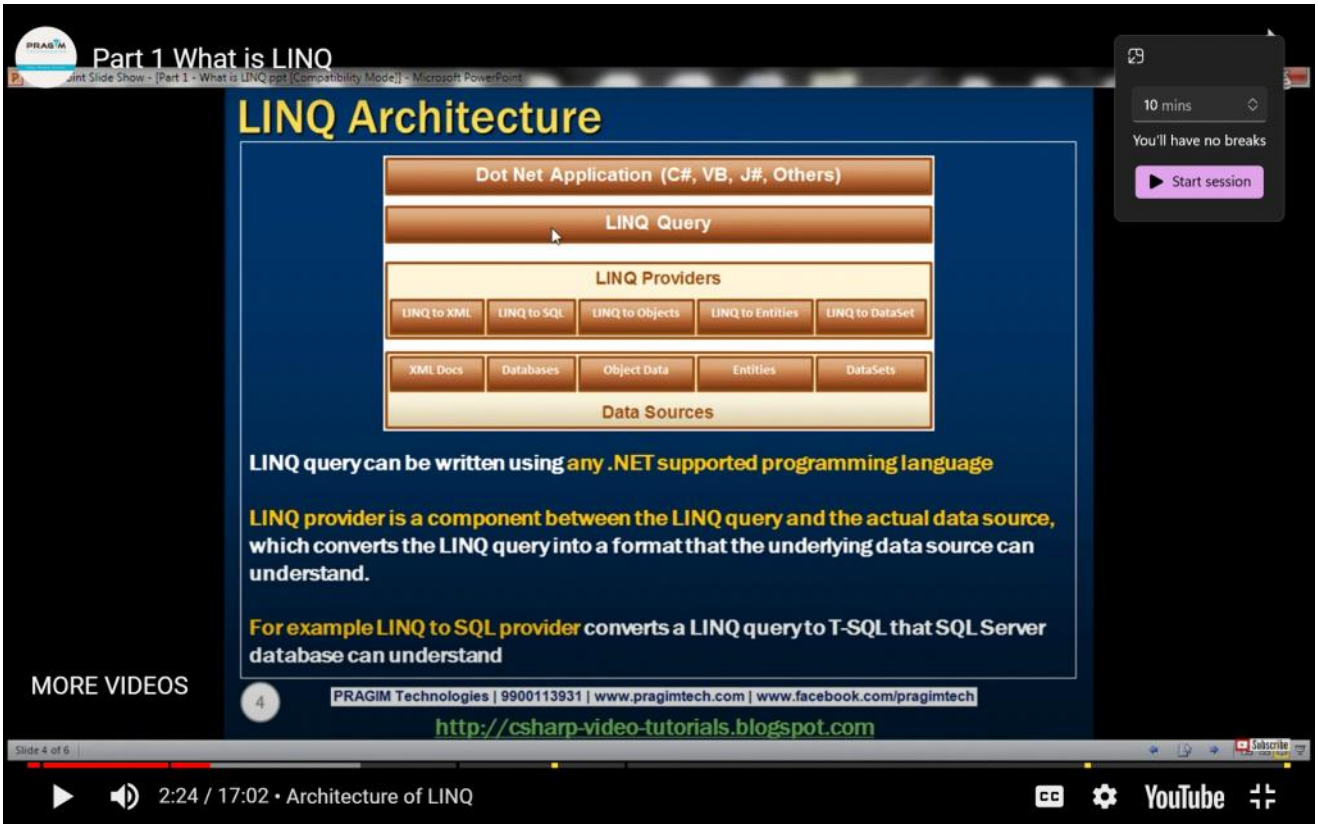


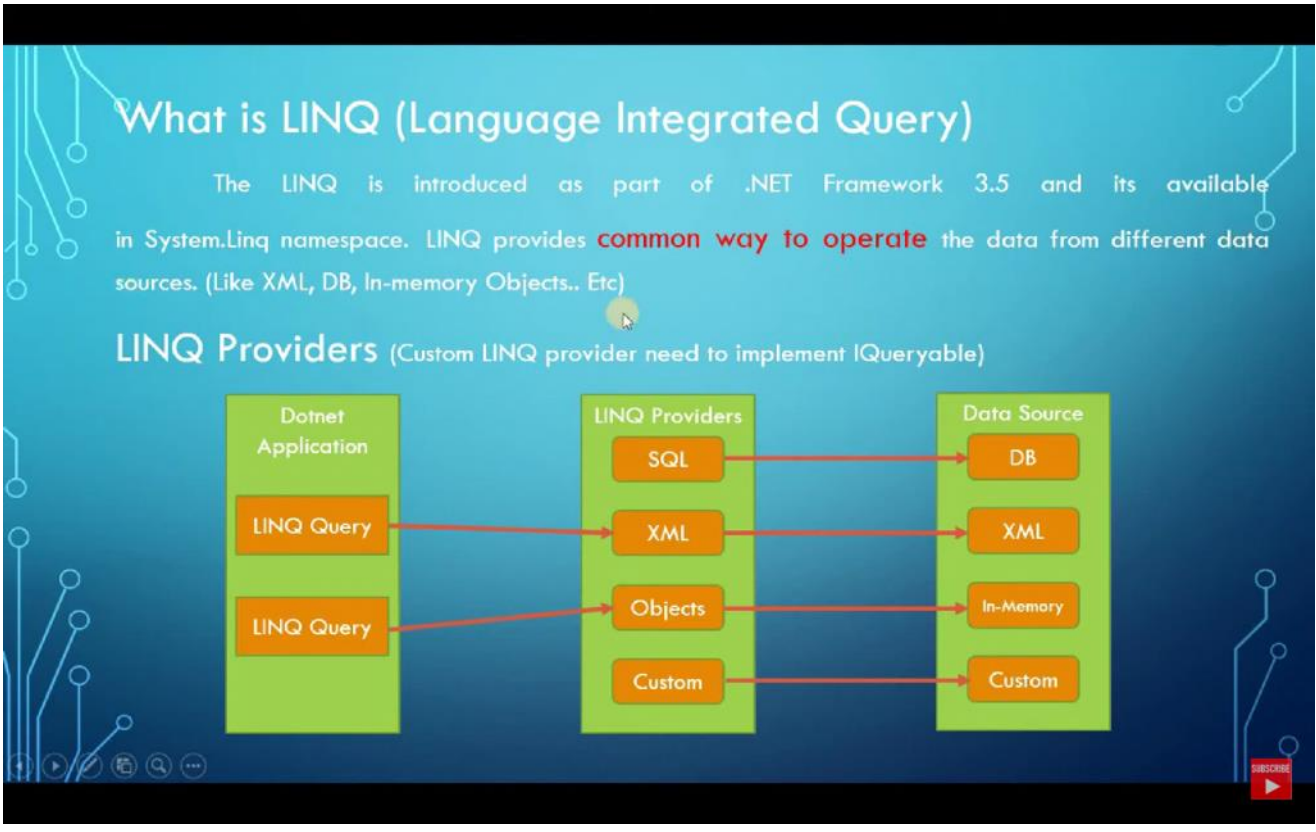
LinQ Stands for Language Integreted Query .

Integerated with any type of database stores like SQL Server ,XML Documents ,Databases

Linq Providers :

Linq providers are the capacity and responsibility of the linq queries to sql or database services .


Part 1 What is LINQ
LINQ Architecture
Dot Net Application (C#, VB, J#, Others)
LINQ Query
LINQ Providers
LINQ to XML
LINQ to SQL
LINQ to Objects
LINQ to Entities
LINQ to DataSet
XML Docs
Databases
Object Data
Entities
DataSets
Data Sources
LINQ query can be written using any .NET supported programming language
LINQ provider is a component between the LINQ query and the actual data source, which converts the LINQ query into a format that the underlying data source can understand.
For example LINQ to SQL provider converts a LINQ query to T-SQL that SQL Server database can understand
MORE VIDEOS
PRAGIM Technologies | 9900113931 | www.pragimtech.com | www.facebook.com/pragimtech
http://csharp-video-tutorials.blogspot.com
10 mins
You'll have no breaks
Start session
2:24 / 17:02 • Architecture of LINQ
YouTube


What is LINQ (Language Integrated Query)
The LINQ is introduced as part of .NET Framework 3.5 and its available in System.Linq namespace. LINQ provides common way to operate the data from different data sources. (Like XML, DB, In-memory Objects.. Etc)
LINQ Providers (Custom LINQ provider need to implement IQueryable)
Dotnet Application
LINQ Query
LINQ Query
LINQ Providers
SQL
XML
Objects
Custom
Data Source
DB
XML
In-Memory
Custom

Aggregate Operators

Monday, February 5, 2024 11:03 AM

Min Max Count Avg

Partitioning Operators

Monday, February 5, 2024 4:46 PM

Take :

The Take operator is used to select the first N elements from a sequence. It's useful when you only need a specific number of elements from the beginning of a collection.

```
var numbers = new List<int> { 1, 2, 3, 4, 5 };
var firstThree = numbers.Take(3); // Results in { 1, 2, 3 }
```

Skip :

The Skip operator is used to bypass a specified number of elements in a sequence and then returns the remaining elements. It's useful for skipping over a known number of elements that you're not interested in.

```
var nextTwo = numbers.Skip(3); // Results in { 4, 5 }
```

TakeWhile :

The TakeWhile operator returns elements from the start of a sequence as long as a specified condition is true. Once an element is encountered that does not satisfy the condition, the process stops, and no further elements are returned.

```
var numbers = new List<int> { 1, 3, 5, 7, 4, 9 } ;
var takeWhileLessThanSix = numbers.TakeWhile(n => n < 6); // Results in { 1, 3, 5 }
```

SkipWhile :

Conversely, the SkipWhile operator skips elements from the sequence as long as the specified condition is true. Once an element fails the condition, the rest of the sequence is returned, including the element that does not satisfy the condition.

```
var skipWhileLessThanSix = numbers.SkipWhile(n => n < 6); // Results in { 7, 4, 9 }
```

CRM

Wednesday, February 7, 2024 9:24 AM

What is CRM ?

CRM stands for Customer Relationship Management. It is a technology for managing all your company's relationships and interactions with current and potential customers. The goal is simple: Improve business relationships to grow your business. A CRM system helps companies stay connected to customers, streamline processes, and improve profitability.

CRM=> CENTRALIZING ALL CUSTOMER DATAS TO USE IN EVERY DEPARTMENT

Types :

- **Operational CRM systems.**
- **Analytical CRM systems.**
- **Collaborative CRM systems.**
-

- **Reaching a potential customer.**
- **Customer acquisition.**
- **Conversion.**
- **Customer retention.**
- **Customer loyalty.**

What is CRM?

CRM stands for Customer Relationship Management. Imagine if you had a friend, and you wanted to make sure they felt happy and important every time they talked to you. You'd remember their birthday, what they like, and what they don't like, right? CRM is like doing that, but for businesses and their customers. It's a way for businesses to keep track of all the information and conversations they have with their customers, so they can make sure the customers are happy and keep coming back.

Why Use CRM?

Businesses use CRM for a few big reasons:

- **To understand their customers better: Like remembering a friend's favorite ice cream flavor, businesses use CRM to remember what their customers like and don't like.**
- **To provide better service: It helps them solve customers' problems faster and more effectively.**
- **To sell more: By knowing what their customers need, businesses can suggest products or services that fit just right.**

How Does CRM Work?

Think of CRM as a giant electronic notebook that keeps all the information about every customer:

- **Contact details: Like names, phone numbers, and addresses.**
- **Conversations: Notes from phone calls, emails, or chats.**
- **Preferences and history: What they've bought, what they looked at on the website, and what they've said they like or need.**

This "notebook" is used by different parts of the business, like the sales team, customer service, and marketing, so everyone knows what's going on with each customer.

When is CRM Used?

CRM is used throughout the entire relationship with a customer:

- **Before a sale: To keep track of potential customers and what they might be interested in.**
- **During a sale: To manage the details of the transaction and make sure everything goes smoothly.**
- **After a sale: To provide support, follow up, and keep the customer engaged with new offers or products.**

Where is CRM Used?

CRM is used in all kinds of businesses, big and small, from a local bakery that keeps track of regular customers' favorite treats, to huge international companies managing millions of customer interactions.

How Really Does CRM Work?

Imagine a toy box, but instead of toys, it's filled with pieces of information about customers. Here's how CRM sorts through that toy box:

1. **Collecting Information:** **Every time a customer interacts with the business, it's like adding another toy to the box.**
2. **Organizing Information:** **CRM helps sort through the box to find exactly what you're looking for, like sorting toys into ones that are cars, dolls, or blocks.**
3. **Using the Information:** **When a business wants to make a customer happy, it looks in the toy box to find exactly what will help, like picking the perfect toy to play with.**

This process helps businesses make better decisions, improve customer service, and sell more effectively.

CRM, or Customer Relationship Management, is a strategy for managing an organization's relationships and interactions with potential and current customers. It helps businesses stay connected to customers, streamline processes, and improve profitability. CRM systems and applications support this strategy by allowing businesses to manage customer interactions, data, and processes in a centralized location. There are several types of CRM systems, each focusing on different aspects of customer relationships:

1. **Operational CRM: This type focuses on automating and improving business processes related to customer interactions, such as sales, marketing, and service. Operational CRM systems offer tools for lead management, contact management, email marketing, and customer service support, aiming to generate leads, convert them into contacts, capture all required details, and provide service throughout the customer lifecycle.**

2. **Analytical CRM: Analytical CRM systems focus on analyzing customer data and behaviors to make informed business decisions. They help organizations understand customer preferences, trends, and patterns by analyzing data from various channels and touchpoints. This type of CRM is crucial for segmentation, targeting, personalization, and improving overall customer satisfaction.**

3. **Collaborative CRM: This type aims to share customer information among various business units such as sales, marketing, and support teams, as well as external stakeholders like suppliers or distribution partners. The goal is to improve service and ensure customers have consistent experiences across all touchpoints. Collaborative CRM focuses on integration and data sharing to enhance teamwork and coordination.**

4. **Strategic CRM: Strategic CRM is centered around the integration of customer**

information into business strategies and operations to improve and maintain customer relationships for long-term growth. It involves analyzing customer data to define policies and processes that prioritize customer needs and relationship building at every organizational level.

How analytical CRMs work :

analytical CRMs work best for high-level strategizing. Data analysis is how you take all the customer information you've collected over time and start answering questions with it.

Analytical CRMs provide reporting features that help you understand:

- **What specific marketing campaigns generate the most leads**
- **What kinds of leads most often turn into sales**
- **What types of sales actions lead to a purchase**
- **Which types of customers have the highest lifetime value**
- **What issues customers most often contact support about**
- **The most frequent customer complaints**
- **What features and resources customers use and like the most**
- **How effectively your support team resolves customer issues**
- **How quickly your agents achieve resolution**

Analytical CRMs are good for:

Companies that have a lot of customer data and don't know how to effectively use it.

How collaborative CRMs work

CRMs tend to be geared more toward customer retention and satisfaction than making sales

With collaborative CRMs, you see a few main features:

- **Interaction management.** A collaborative CRM makes it easy to track every interaction a customer or prospect has with your company, no matter the channel. The information in the product gets updated whether a customer got in touch via phone, email, social media, messaging apps, or even through an in-person meeting with someone at the company. Agents can record what the interaction was about, how it resolved, and add any important details someone might need to know for a future interaction with the customer.

- **Channel management.** Customers now expect brands to be available across multiple channels. It's important to understand which channels your customers prefer, and figure out the best ways to meet them where they are. Collaborative CRMs help track which channels your customers are using for different types of contact, to ensure you're available when they need you where they want you.

- **Document management.** Some collaborative CRMs also help companies consolidate where they store important customer documents. If employees may need access to a customer

contract or proposal in the course of helping a customer out, this feature can come in handy.

How operational CRMs work

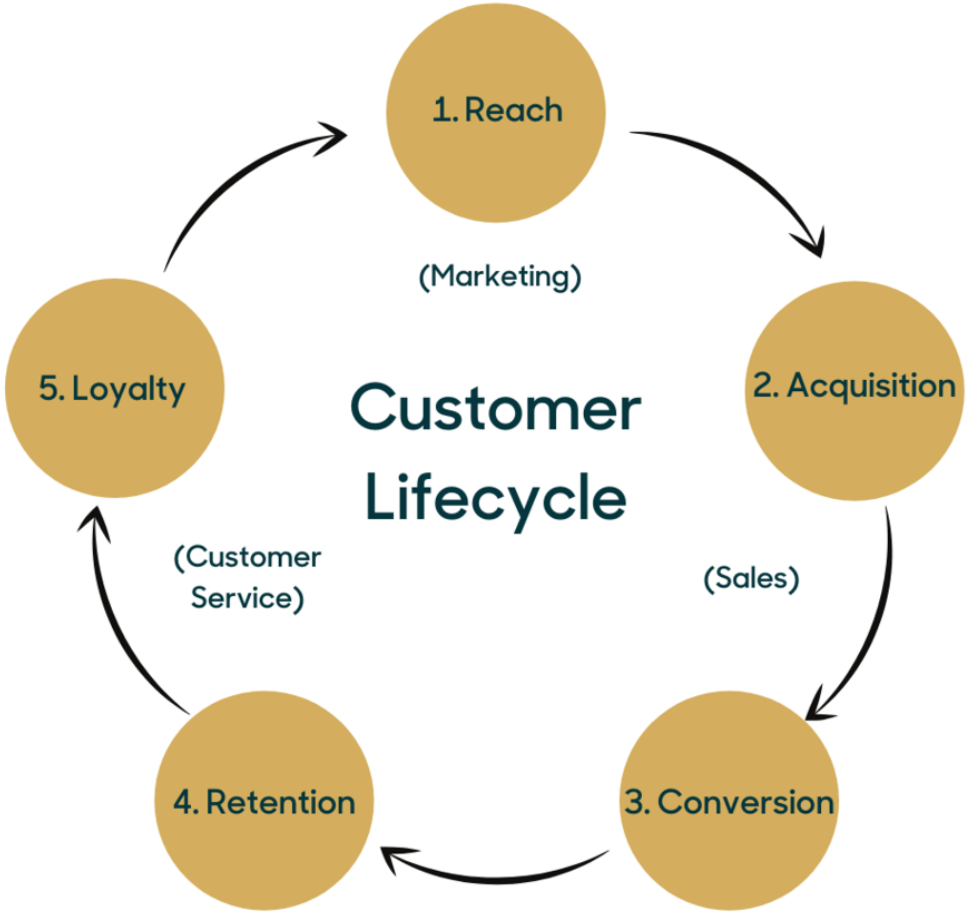
And operational CRMs are where automation features start to come more into play. In order to bring greater efficiency to all the processes related to managing customer relationships, operational CRMs frequently include features for sales automation, marketing automation, and service automation.

Operational CRMs are good for:

Businesses that want to get more out of the customer information they have, while making processes more efficient for employees. And businesses that want to gain a high-level view of the entire customer lifecycle and find ways to make your processes across customer-facing departments better.

Can i use all three crms at the same time ?

Yes, you absolutely can. In fact, many businesses prefer to use multiple types of CRM systems so that they can improve the efficiency of their business processes and accomplish more things at once.



zendesk sell

Automation

Wednesday, February 7, 2024 5:30 PM

Automation:

Automation is the use of technology to perform tasks with reduced human assistance.

Imagine the factory has robots that can make toys all by themselves! You tell them once what to do, like "make a teddy bear," and they keep making teddy bears without needing to be told again. This helps the factory make lots of toys quickly without everyone having to do everything by hand.

What is Automation ?

Automation is the process of creating software and systems to replace repeatable processes and reduce manual intervention.

Why Is Automation ?

automation can increase productivity and efficiency within your business. It can also deliver new insights into business and IT challenges and suggest solutions using rules-based decisioning.

The benefits of automated operations are higher productivity, reliability, availability, increased performance, and reduced operating costs.

Where is Automation really needs ?

Automation is used in a wide variety of fields and industries to improve efficiency, accuracy, and productivity.

Automation in CRM :

Automation in Customer Relationship Management (CRM) software focuses on enhancing customer engagement, streamlining sales processes, and improving service efficiency. It leverages technology to automate repetitive tasks, manage customer data, and personalize customer interactions. Here's how automation is applied in CRM:

CRM Automation and It's Types :

Customer Relationship Management (CRM) automation involves using software to automate repetitive tasks and processes related to managing customer relationships. This not only saves time but also enhances customer engagement and personalizes the customer experience. CRM automation can be categorized into several types, each focusing on different aspects of customer relationship management:

1. Marketing Automation

This type involves automating marketing processes and campaigns to efficiently manage leads and nurture them until they are sales-ready. Key features include:

- **Email Campaigns: Automated sending of personalized emails based on customer actions or time triggers.**
- **Lead Scoring: Automatically scoring leads based on their engagement and behavior to prioritize sales efforts.**
- **Customer Segmentation: Automatically segmenting customers based on their demographics, behavior, and preferences for targeted marketing efforts.**

2. Sales Automation

Sales automation streamlines the sales process, from lead capture to closing deals, by automating tasks that are repetitive and time-consuming. Key features include:

- **Lead Management: Automating the capture, tracking, and nurturing of leads.**
- **Contact and Deal Management: Automatically updating customer information and managing deals through the sales pipeline.**
- **Task Automation: Scheduling tasks and reminders for follow-ups, meetings, and calls to ensure nothing falls through the cracks.**

3. Customer Service Automation

This type focuses on automating aspects of customer service to provide quick and consistent support. Key features include:

- **Ticketing System: Automatically creating and assigning support tickets based on customer inquiries.**
- **Chatbots and Virtual Assistants: Providing immediate automated responses to**

common customer queries.

- **Knowledge Base Management:** Automatically updating and managing a self-service portal for customers.

4. Analytics and Reporting Automation

Automating the generation of reports and analytics helps businesses gain insights into customer behavior, sales trends, and marketing effectiveness. Key features include:

- **Sales Reporting:** Automatically generating sales reports to track performance and identify trends.
- **Customer Analytics:** Analyzing customer data to understand behavior patterns and preferences.
- **Marketing Performance:** Measuring the effectiveness of marketing campaigns and activities.

5. Workflow Automation

This involves automating internal processes and workflows to ensure smooth operations across different departments. Key features include:

- **Process Management**: Automating business processes and workflows to improve efficiency.
- **Approval Processes:** Streamlining the approval process for discounts, contracts, and more through automation.
- **Data Synchronization:** Automatically syncing data across different systems and platforms to maintain consistency.

6. Social Media Automation

Automating social media interactions and engagements to maintain an active and responsive online presence. Key features include:

- **Social Listening:** Automatically monitoring social media for mentions, keywords, or sentiments related to the brand.
- **Post Scheduling:** Automating the scheduling and posting of content on various social media platforms.
- **Engagement Tracking:** Tracking and analyzing engagement metrics to inform strategy.

Sale Automation :

Sales automation refers to the use of software to automate repetitive and time-consuming tasks involved in the sales process. This enables sales teams to focus more on engaging with potential customers and closing deals rather than being bogged down by administrative tasks.

Key Components of Sales Automation

1. Lead Capture and Management: Automatically captures leads from various sources like websites, social media, and email campaigns, and organizes them in a centralized database. It can also classify leads based on predefined criteria, making it easier for sales teams to prioritize and manage them.
2. Email Automation: Automates the sending of personalized emails and follow-ups based on specific triggers, such as a lead downloading a brochure or visiting a particular webpage. This ensures timely communication with prospects without manual intervention.

3. **Activity Tracking: Keeps track of all interactions with leads and customers, including emails, calls, meetings, and social media engagements. This information is automatically logged into the CRM system, providing a comprehensive view of each customer's journey.**
4. **Task Automation: Automatically assigns tasks to sales representatives based on specific triggers or rules. For example, a follow-up task can be created when a lead opens an email but doesn't respond within a certain timeframe.**
5. **Sales Pipeline Management: Automates the movement of leads through different stages of the sales funnel, from initial contact to closing the deal. It can notify sales reps when leads progress to a new stage or when action is required to move a deal forward.**
6. **Reporting and Analytics: Generates automatic reports on sales performance, lead conversion rates, and other key metrics. This helps sales teams and managers make data-driven decisions to improve their strategies.**

Workflow Automation :

Workflow automation in Customer Relationship Management (CRM) refers to the use of software to automate repetitive and manual tasks within the CRM system to improve efficiency, accuracy, and consistency. It involves setting up rules and triggers within the CRM platform that automatically initiate actions or processes based on specific criteria or events.

Key Components :

Triggers :

These are events that initiate an automated workflow.

For example ,
a new lead entry in the CRM can trigger an automated welcome email.

Conditions:

These specify criteria that need to be met for the workflow to proceed.

For instance, a condition could be that the lead must be marked as "qualified" before an automated task for a sales call is created.

Actions:

These are the tasks or processes that are automatically executed when triggers and conditions are met.

Actions can include sending emails, updating contact records, assigning tasks to team members, or even initiating a series of follow-up activities.

Why?

Because people can get really busy, and sometimes they forget to do things like sending a thank you note when someone buys a toy (product or service) or wishing them on their birthday. Automation helps make sure these things don't get forgotten.

What is Lead Nurturing ?

At its core, lead nurturing is the process of cultivating leads that are not yet ready to buy.

What is it?

Workflow automation in CRM is like having a smart robot that knows when to do things like send out thank you cards, remind you of someone's birthday, or even sort your toys based on which ones you play with the most. It helps make sure customers are happy without someone having to remember to do all these things.

When will it be?

It works all the time! Once someone sets it up, the automation keeps working day and night, making sure all the tasks are done at the right time, just like how a night light turns on automatically when it gets dark.

How is it used?

Let's say you make a new friend, and you want to remember to send them a card on their birthday. Your mom can put a reminder in her phone, and when the time comes, it tells her to send the card. In CRM, it's similar; the system is set up to automatically send messages or do tasks based on what people need or what time it is, like sending an email to customers when they sign up for something new.

Why?

It's all about making sure customers feel special and taken care of, without making people do all the work by hand. It's like how you feel happy when you find your toys exactly where you expect them to be, customers feel happy when businesses remember important things about them and make things easier.

AUTO ASSIGNMENT AUTOMATION :

Auto assignment automation in Customer Relationship Management (CRM) systems refers to the process of automatically allocating leads, contacts, customer inquiries, support tickets, or tasks to the appropriate team members, sales representatives, or service agents based on predefined rules, criteria, or algorithms. This feature aims to improve efficiency, response times, and customer satisfaction while ensuring a balanced workload among staff. Here's an overview of how it works

strategic approach to streamline and optimize the distribution of leads, tasks, inquiries, or support tickets among team members.

WEB HOOKS :

Webhooks in the context of Customer Relationship Management (CRM) systems are an essential feature for enabling real-time data synchronization and automated workflows between the CRM and other applications. They function as user-defined HTTP callbacks, which are triggered by specific events within the CRM system, such as the creation of a new lead, a change in a customer's status, or the completion of a sales transaction.

Real-Time Example of Webhooks in CRM Automation

Lead Capture and Immediate Follow-Up:

- **A potential customer fills out a contact form on the company's website, which is captured as a new lead in the CRM.**
- **A webhook is triggered by the new lead creation event. It sends the**

lead's contact information to the email marketing software.

- The email marketing software automatically enrolls the lead in a nurturing campaign, sending them a welcome email and subsequent communications based on their interactions.

Customer Support Ticket Automation:

- The same customer, now engaged through the nurturing campaign, replies to an email with a query about the product.
- This email interaction is captured in the CRM, and a specific keyword within the customer's reply triggers another webhook.
- The webhook sends a request to the customer support ticketing system with the customer's query, automatically creating a new support ticket.
- The customer support team receives the ticket and can immediately address the customer's question, enhancing the customer experience.

SLA IN CRM AUTOMATION :

Service Level Agreements (SLAs) in Customer Relationship Management (CRM) automation play a crucial role in setting clear expectations for service delivery between businesses and their customers. SLAs define the standards for response times, resolution times, and the overall quality of service that a business commits to providing its customers. Integrating SLAs into CRM automation helps ensure that these commitments are consistently met through automated tracking, notifications, and reporting.

Technology Used

Friday, February 9, 2024 5:56 PM

Identity Server :

Just like you use your key to open your toy box, your parents use these special keys (like a username and password) to safely enter and use different websites and apps. So, Identity Server is like a guardian that makes sure only the right people can open and use websites and apps, keeping everything safe and sound!

GOOGLE CLOUD SQL :

Google Cloud SQL is like a super-secure, always available digital storage space for all the important information your apps and websites need to work properly.

What is Queueing ?

Imagine you and your friends are waiting in line to slide down your favorite slide at the playground. The first person in line goes down the slide first, and everyone else follows in the order they arrived in line. This is a lot like queueing!

What is Azure Bus Service ?

Imagine you and your friends have walkie-talkies to talk to each other. But sometimes, you want to send a message when your friend isn't listening, or maybe you want to talk to many friends at once without repeating the same thing over and over. Azure Service Bus is like a magical walkie-talkie station that helps with this.
Think of Azure Service Bus as a big, invisible mailbox in the sky. When you want to tell your friend something, but your friend is busy or away, you can leave your message in this sky mailbox. Your friend can check the mailbox later and get your message, even if you're not there anymore.
And if you want to tell something to all your friends at once, instead of calling each one separately, you can just send one message to the sky mailbox. The mailbox knows who your friends are and delivers your message to all of them. So, everyone gets the message without you having to repeat yourself.
This magical mailbox makes sure nobody misses any messages, and it can hold onto them until your friends are ready to listen.

It's like having a superpower for your messages, making sure they always reach your friends, whether they're ready right now or later.

What is AWS ?

Imagine you have a giant, magical toy box that can give you any toy you want to play with, anytime and anywhere. This toy box is very special because it can also grow bigger when you need more toys and shrink smaller when you want fewer toys, so it always fits perfectly in your room.
AWS, or Amazon Web Services, is like this magical toy box, but for computers and the internet. Instead of toys, AWS gives people and companies all sorts of computer things they need, like space to store data, powerful computers to help them do big tasks, and even special tools to make websites and apps work better.

Hosting :

Hosting, in the context of the internet, is a service that allows individuals and organizations to make their website accessible to the world via the World Wide Web. It's like renting a space on a computer server where your website's files are stored. This server is always connected to the internet, so anyone can access your website at any time from anywhere in the world.

Caching :

Imagine you have a favorite book that you love reading over and over again. Instead of going to the library every time you want to read it, you keep it in your room. That way, it's much easier and faster for you to grab the book and start reading whenever you want.
In the world of computers and the internet, caching works in a similar way. When you visit websites or use apps, your computer (or another device) saves some information from those websites or apps, like pictures or parts of web pages. The next time you go to the same website or use the same app, your computer can quickly show you the saved information without having to go out and get it all over again from the internet. This makes everything load faster, so you don't have to wait as long to see websites or use apps. Caching is like keeping your favorite book in your room for quick access, but for digital information.

LOG :

Imagine you're a detective with a notebook. Every time you discover a clue, you write it down in your notebook. Over time, your notebook fills up with notes about all the clues you've found and the things you've done during your investigations. This helps you remember what happened and when, so if you ever need to go back and check something, you can just look in your notebook.
In the world of computers, "logs" are like the detective's notebook. They are files that keep track of what the computer or a computer program does. Every time something happens, like if you visit a website or if there's an error when you're trying to do something, the computer writes a note about it in its logs. These notes include information like what happened, when it happened, and sometimes why it happened. People who take care of computers and websites use logs to help them understand how things are working. If there's a problem, they can look at the logs to find clues and figure out what went wrong, just like a detective looks back through their notebook. This helps them fix problems and make sure everything runs smoothly.

EXCEPTION :

Imagine you're playing a game where you're supposed to follow certain rules, like only walking on a path or only jumping over obstacles that are a certain color. Now, suppose you come across something unexpected that doesn't fit the rules, like a big puddle in the middle of your path or a new obstacle that's a color you've never seen before. This surprise or "exception" means you have to figure out a different way to move forward because the usual rules don't apply.
In computer programming, an "exception" is a lot like that unexpected puddle or strange-colored obstacle. It's something that happens when the computer is following a set of instructions (a program) and encounters a situation it doesn't know how to handle with its current set of rules. This could be anything from trying to read a file that doesn't exist, to running out of memory, or trying to do a math problem that doesn't make sense (like dividing by zero).
When a program comes across an exception, it needs to figure out what to do next. Programmers write special parts of the program called "exception handlers" to deal with these unexpected situations. These handlers are like instructions that tell the program, "If you hit a problem you can't handle the normal way, here's what to do instead." This helps make sure the program can keep running smoothly, or at least stop in a way that's safe and tells the user what went wrong, instead of just crashing or freezing up.

CONTINUOUS INTEGRATION :

Imagine you and your friends are building a giant puzzle together, but instead of doing it all at once in the same place, each of you is working on small parts of the puzzle at your own homes. Every time someone finishes a part, they bring it over to fit it into the big puzzle. If the new piece doesn't fit right, you figure out what went wrong immediately and fix it before adding more pieces.
Continuous Integration (CI) is a lot like working on this puzzle together. It's a practice in software development where developers frequently merge their code changes into a central repository, often multiple times a day. Each time new code is added, it's automatically tested to make sure it works well with everything else that's already there. This is like checking if the new puzzle piece fits with the rest of the puzzle.
By doing this, developers can find and fix problems early, before they grow into bigger issues. It helps everyone in the team to stay on the same page and makes sure that the final software product is as error-free as possible. Continuous Integration helps in building the software piece by piece, ensuring that each new piece fits perfectly with the others, making the whole process more efficient and the final product more

INTEGRATION TESTING :

Integration Testing in the world of computer programming is a bit like this race. Developers write different parts of a software program separately, like how you and your friends built your cars from different sets. Each part works well by itself, which is checked during something called "unit testing."
Once all these parts are ready, integration testing is the next step. It's where developers put all these different parts together and test them as a group. They're checking to see if the parts communicate with each other correctly, like making sure your toy cars can race together without issues. They look for problems where the parts connect or interact, such as data being passed incorrectly between them or processes not working as they should when combined. This helps ensure that when all parts of the program are put together, they work smoothly as a whole, just like ensuring all the toy cars can complete the race on the track together.

UI TESTING ?

Imagine you've built a really cool robot with buttons on it to make it walk, turn, and even dance. You want to make sure that when you or your friends press the buttons, the robot does exactly what it's supposed to do. So, you test it by pressing each button and watching to see if the robot walks, turns, or dances like you expect. This testing is to make sure everyone can easily use the robot and enjoy playing with it.
UI (User Interface) testing in the world of computers is similar to checking the robot's buttons. A software program or an app has a "UI" - which is like its face or the buttons on your robot. It's what people see and interact with on the screen, like buttons, menus, and icons. UI testing is when developers check to see if all these visual and interactive parts work correctly. They make sure that when you click a button or enter information into a form on a website or an app, it does exactly what it's supposed to do.
Just like how you want your robot to respond correctly to button presses, UI testing ensures that software responds correctly to user actions. This helps make sure that people can use the software easily and have a good experience without running into problems like buttons that don't do anything or menus that are hard to navigate.

Scheduler

Friday, February 9, 2024 3:30 PM

A Scheduler is a tool or program that helps computers, phones, or other smart devices know when to do certain tasks all by themselves. For example, a Scheduler can tell a computer to check for new emails every hour, save your work automatically so you don't lose it, or even start a robot vacuum to clean the floor at the same time every day.

What is rules creation

Friday, February 9, 2024 3:15 PM

Imagine you have a set of toy cars and you're organizing a race. Before the race starts, you decide on rules like "No pushing other cars off the track" or "You must go around the track three times to finish." These rules help make the race fair and fun for everyone.
In a similar way, "rules creation" in technology, games, or any system is like setting up those race rules. It's about deciding what should happen when certain things are done. For example, in a video game, a rule might be "If the player collects 100 coins, they get an extra life." This tells the game what to do when you reach 100 coins.
In computers or apps, rules can help with lots of things like sending you an email when someone fills out a form on your website, or adding points to a loyalty card when a purchase is made. These rules make sure that everything works smoothly and the way it's supposed to, just like the rules of the toy car race make sure the race is fair and fun.

MICRO SERVICES

Friday, February 9, 2024 2:31 PM

Microservices are a way of building software where instead of making one big program (like the giant castle), you make lots of small programs (like the small LEGO sets). Each small program does one thing really well, like managing user logins, handling orders, or keeping track of what's in your shopping cart .

These small programs can talk to each other, but they work independently. This means if you need to fix something or add something new, you can just work on the small part that needs it without having to deal with the whole big program. It's like being able to change the gate of your castle without having to rebuild the whole thing.

So, Microservices make it easier for people who make software to manage big, complex programs by breaking them down into smaller, easier-to-handle pieces.

Transaction Table :

Frequent ah update aagikittee irukum

Automation rule uh, automation record uh, automation pipleline mapper , automation rule log , automation action log ,

Master Table :
Others

Frequent uipdate nadakkkathu…

workflow automation ethukaga panrom
2. Azure bus na enna
3. time trigger epdi nadakuthu
4.Instant na enna delayed na enna
5. etho oru two Tables pathi kekanu
4. Transaction tables ennana

workflow automation ethukaga panrom
2. Azure bus na enna
3. time trigger epdi nadakuthu
4.Instant na enna delayed na enna
5. etho oru two Tables pathi kekanu
4. Transaction tables ennana

Azure Bus Service

Friday, February 9, 2024 2:24 PM

Imagine you and your friends have walkie-talkies to talk to each other. But sometimes, you want to send a message when your friend isn't listening, or maybe you want to talk to many friends at once without repeating the same thing over and over. Azure Service Bus is like a magical walkie-talkie station that helps with this.
Think of Azure Service Bus as a big, invisible mailbox in the sky. When you want to tell your friend something, but your friend is busy or away, you can leave your message in this sky mailbox. Your friend can check the mailbox later and get your message, even if you're not there anymore.
And if you want to tell something to all your friends at once, instead of calling each one separately, you can just send one message to the sky mailbox. The mailbox knows who your friends are and delivers your message to all of them. So, everyone gets the message without you having to repeat yourself.
This magical mailbox makes sure nobody misses any messages, and it can hold onto them until your friends are ready to listen. It's like having a superpower for your messages, making sure they always reach your friends, whether they're ready right now or later.

Azure Service Bus is a cloud messaging service provided by Microsoft Azure that helps applications and services communicate with each other, either within the same cloud environment or across different environments. It acts as a reliable and secure platform for sending and receiving messages between different parts of an application, or between different applications, regardless of whether they are running in the cloud, on-premises, or a combination of both. Service Bus is designed to facilitate complex communication and messaging patterns, making it easier to build applications that are scalable and resilient. Here are some key features and concepts associated with Azure Service Bus:

Key Features

1. **Asynchronous Messaging: Allows applications to send and receive messages without requiring the sender and receiver to be available at the same time, improving efficiency and scalability.**
2. **Decoupled Communication: The sender and receiver of the message do not need to know about each other, which simplifies the architecture and increases flexibility.**

3. **Reliable Delivery: Ensures messages are safely stored until the receiving application is ready to process them, supporting different delivery assurances like At-Most-Once, At-Least-Once, and Exactly-Once.**
4. **Rich Messaging Patterns: Supports a variety of messaging patterns, including simple queues, publish-subscribe (topics and subscriptions), and more complex patterns like dead-letter queues and scheduled messages.**
5. **Scalability and High Availability: Automatically scales to handle high volumes of messages and is designed to provide high availability, ensuring that messages flow smoothly between applications with minimal downtime.**
6. **Security: Offers secure communication through authentication and encryption, protecting messages from unauthorized access.**

Core Components

7. **Queues: Enable one-way communication. A sender sends a message to a queue, and a receiver reads from the queue. It's ideal for point-to-point communication.**
8. **Topics and Subscriptions: Enable publish-subscribe scenarios. A sender publishes a message to a topic, and multiple receivers can receive copies of the message through subscriptions. This is useful for broadcasting messages to multiple receivers.**
9. **Relays: Facilitate hybrid applications by enabling you to securely expose services that run in a corporate network to the public cloud without opening a firewall connection.**

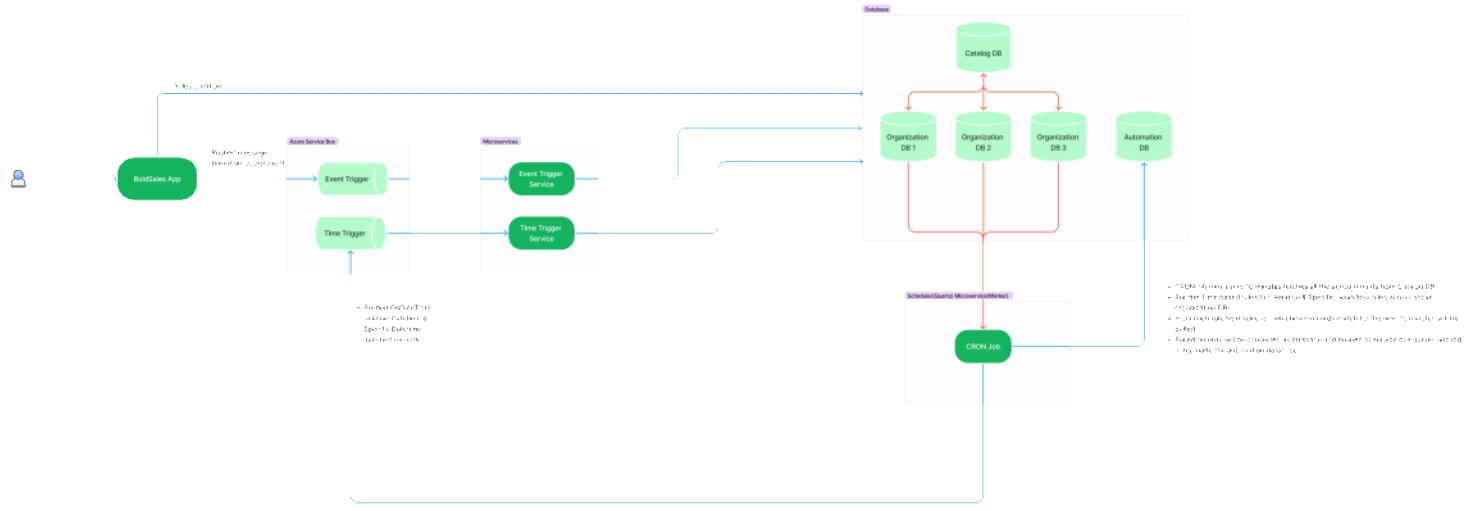
Use Cases

- **Order Processing: Handling orders where the order placement and processing can happen at different times.**
- **Event Notification: Sending notifications to users or systems about events like sales or system updates.**
- **Workflow Processing: Coordinating complex workflows between different parts of an application or across multiple applications.**
- **Load Leveling: Absorbing unexpected spikes in load by queuing up requests that can be processed at a steady rate.**

Azure Service Bus is part of the broader Azure messaging services, which also include Azure Queue Storage for simpler messaging needs and Azure Event Hubs for big data event streaming scenarios. Service Bus is suitable for developers and enterprises that need a robust, scalable service for integrating applications, data, and devices across on-premises and cloud environments.

Architectural Diagram

Friday, February 9, 2024 2:08 PM



Event Triggers

Friday, February 9, 2024 2:20 PM

Imagine you have a magic toy box. This magic box can do special things when certain things happen or at specific times you decide.

Event Trigger:

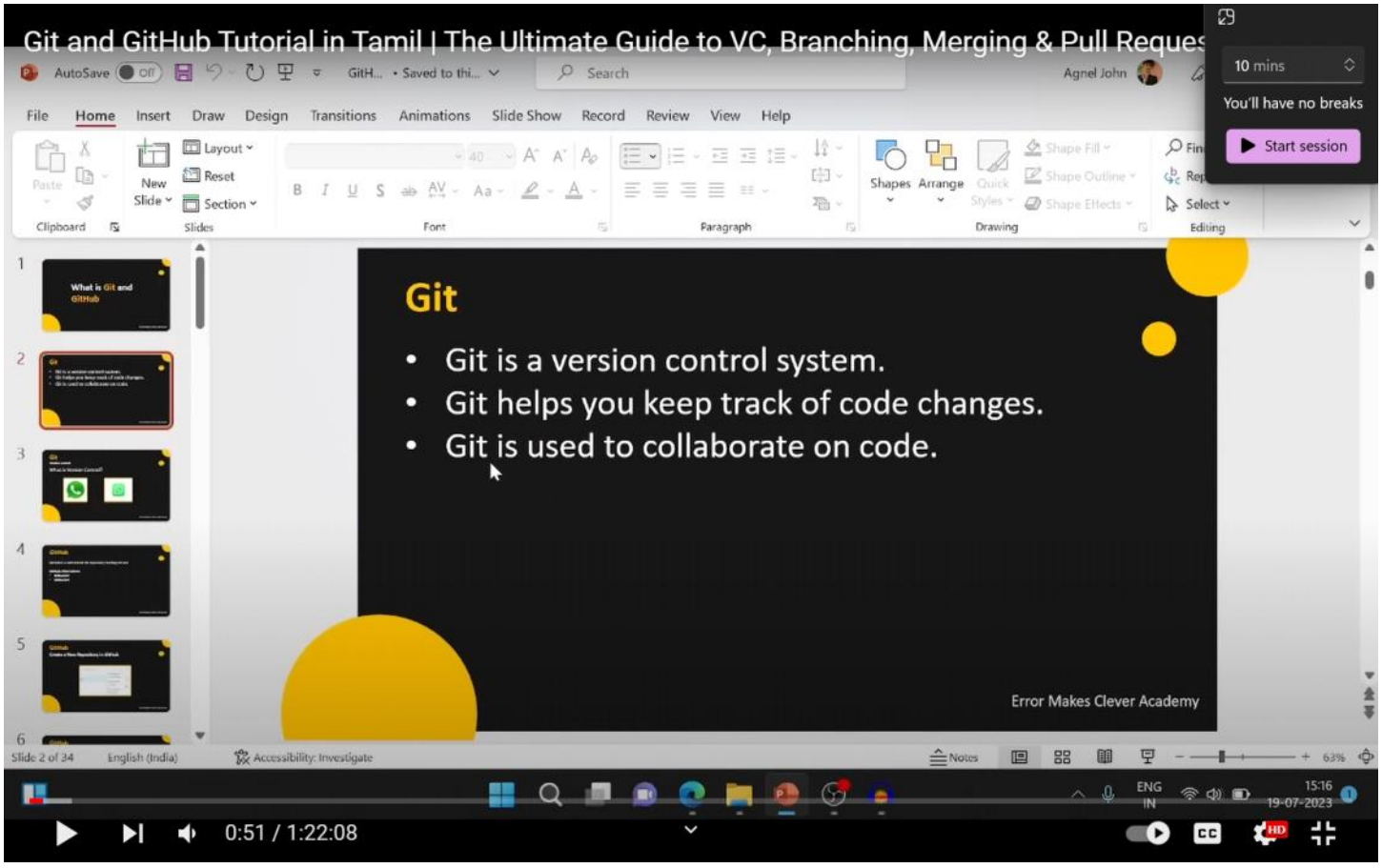
Let's say you have a magic word, "Abracadabra!" Every time you say this word, the toy box opens automatically. That's like an event trigger. In a computer or a special program, an event trigger is like saying "Abracadabra!" It makes the program do something special right after you do a certain thing, like clicking a button or entering some information.

Time Trigger:

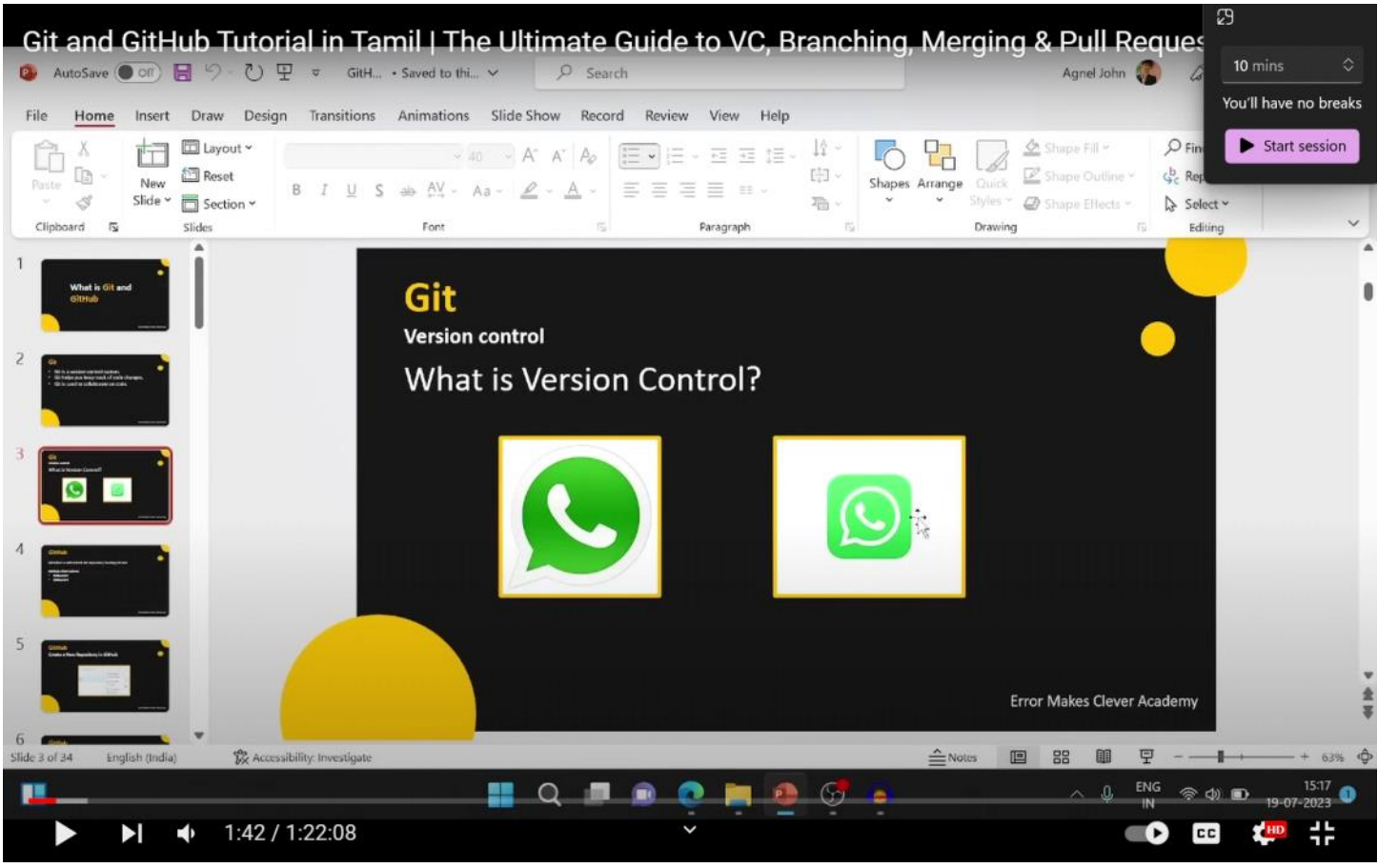
> **Now, imagine you tell your toy box to open every day at 7:00 AM, just in time for you to pick a toy before breakfast. The toy box does exactly that, opens every day at 7:00 AM, without you needing to say "Abracadabra!" That's like a time trigger. In a computer or a program, a time trigger is like setting an alarm clock. It tells the program to do something special at a certain time, like sending a happy birthday message to your friend on their birthday or reminding you about your homework at 4:00 PM every day.**

GIT and GITHUB

Wednesday, February 7, 2024 11:47 AM



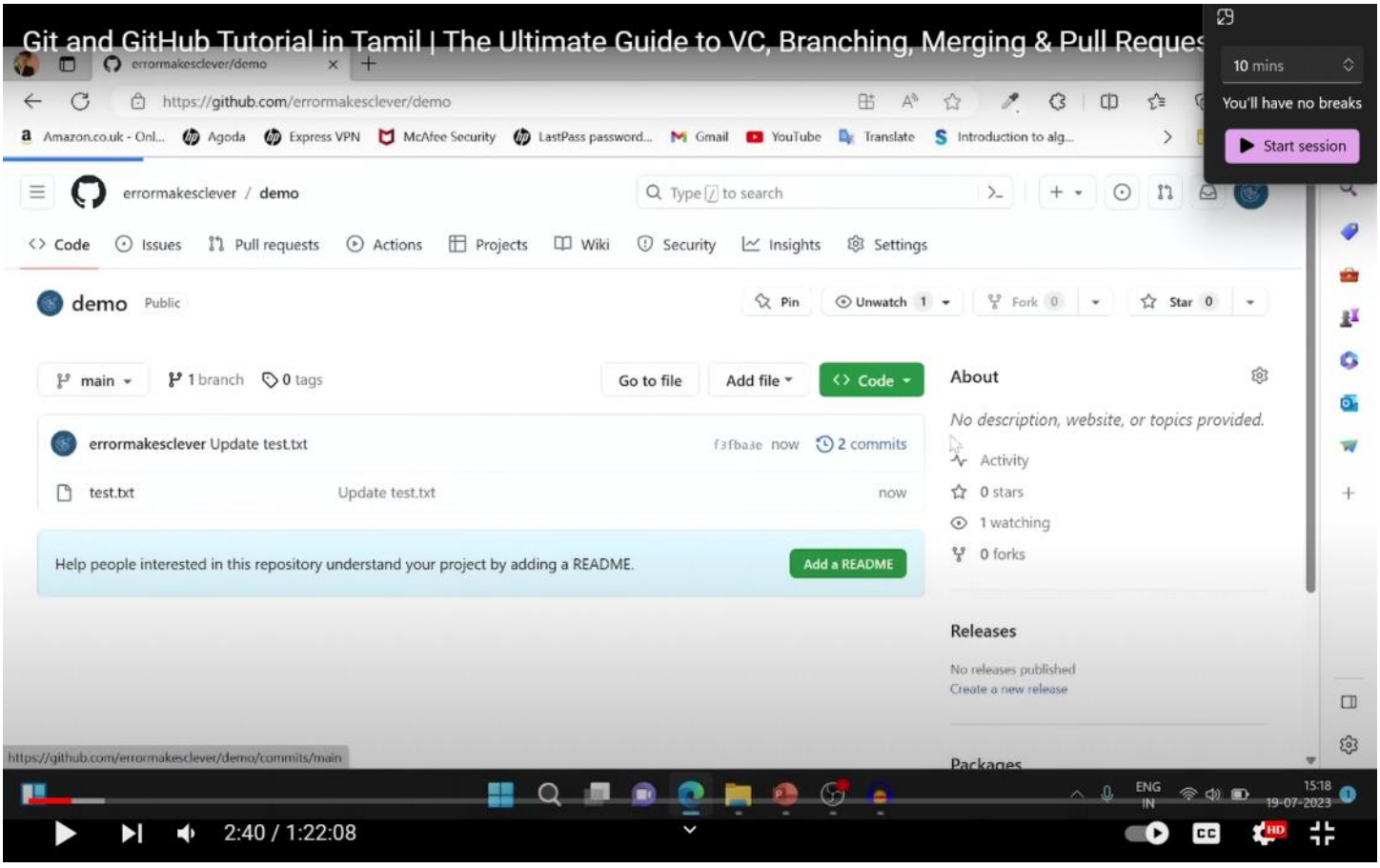
VERSION CONTROL :



HOW TO CHECK THAT DIFFERENT VERSIONS :

To see in the Commit section

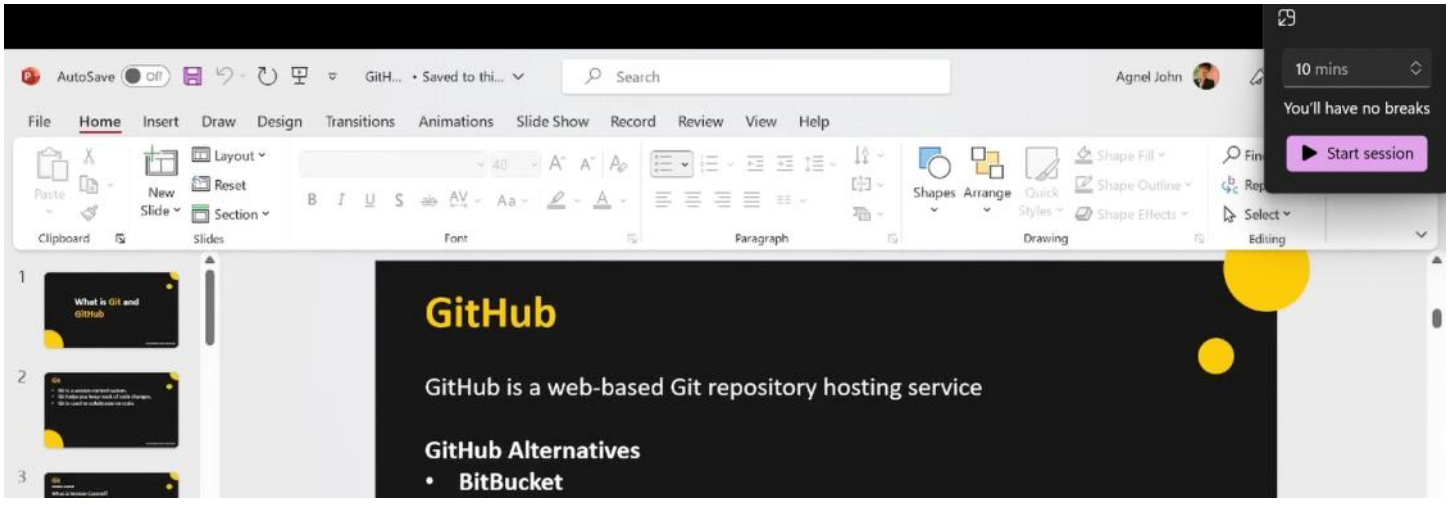
After committing multiple times , we have different commits of versions

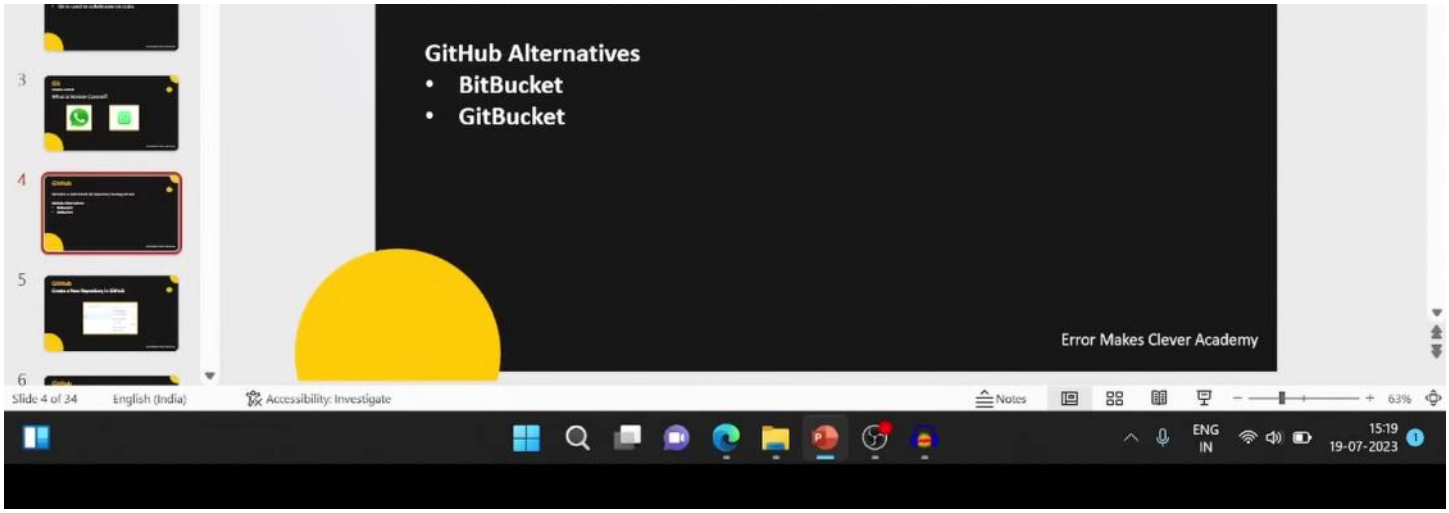


REAL USE CASE :

If the new version is not working or sometimes it might be in the form of malfunctioning , we can just run with previous version .

GITHUB :




GitHub Alternatives
• BitBucket
• GitBucket
Error Makes Clever Academy
Slide 4 of 34
English (India)
Accessibility: Investigate
Notes
63%
ENG
IN
15:19
19-07-2023

Dot Net Core

Tuesday, February 20, 2024 3:37 PM

Fluent Validation

Tuesday, February 20, 2024 3:37 PM

Class file laiye validate pannanumnu avasiyam illaa… decouple pannikkalam

Abstract Validator <T> default class or predefined class . Fluent Validation pannurathuku abstract validator ah inherit pannikkanum .

T refers to enna model ku validation pannaum to that particular class .for example person class

```
public class PersonValidator : AbstractValidator<Person> {

    Public PersonValidator() {

        RuleFor(person => person.Name).NotEmpty().WithMessage("Name is
        required.");
         RuleFor(person => person.Age).InclusiveBetween(0, 100).WithMessage("Age
        must be between 0 and 100."); }
}
```

With State :

Adding extra information

```
public class MyModelValidator : AbstractValidator<MyModel>
{
   public MyModelValidator()
   {
      RuleFor(x => x.SomeProperty)
         .NotNull()
         .WithMessage("SomeProperty must not be null.")
         .WithState(x => new { ErrorCode = "ERR001" });
   }
}
```

Json Serializer

Tuesday, February 20, 2024 4:29 PM

Imagine you have a box of toys, and you want to send it to your friend's house. But there's a problem: the box is too big to fit through the mail slot. So, what you do is take each toy, write down a description of it on a piece of paper (like "red car", "blue ball"), and send these papers through the mail slot. Your friend, who knows what each description means, can then use similar toys from their own collection to recreate the box of toys you wanted to send. This process of writing down descriptions is a bit like what JsonSerializer does in the world of computers.

In computer programming, JsonSerializer is a tool that helps programs to share information. Let's say a program has some data, like a list of your favorite games, but it needs to send this data over the internet or save it in a file. The data, as it exists in the program, isn't in a format that can be easily sent or saved. JsonSerializer takes this data and converts it into a special format called JSON (which stands for JavaScript Object Notation), which is basically a text description of the data. This text can be easily sent over the internet or saved in a file.
Once the data arrives at its destination or needs to be read from a file, JsonSerializer can also do the reverse. It can take the JSON text and convert it back into data that the program can use, recreating the original list of favorite games from the text descriptions.

So, JsonSerializer acts like a translator, turning program data into a format that's easy to share and store, and then back again into a format the program can use.

HTTP STATUS Response Code

Monday, February 26, 2024 12:00 PM

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes