* What is Dependency Injection. What is Scoped, Singleton and Transient

Ans:

<https://www.c-sharpcorner.com/article/understanding-addtransient-vs-addscoped-vs-addsingleton-in-asp-net-core/>

* Method Overriding with example and why we use virtual keyword with function and what is difference between abstract method and virtual method.

Ans:

[https://www.softwaretestinghelp.com/csharp-using-statement-virtual-method/#:~:text=The%20virtual%20keyword%20in%20C%23,is%20preceded%20by%20override%20keyword.](https://www.softwaretestinghelp.com/csharp-using-statement-virtual-method/%23:~:text=The%20virtual%20keyword%20in%20C%23,is%20preceded%20by%20override%20keyword.)

* Different Api method

**HTTP Methods**

* [GET](https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them)
* [POST](https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them)
* [PUT](https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them)
* PUT is used to update the existing resource on the server and it updates the full resource.
* If the resource does not exist, PUT may decide to create a new resource.
* PUT method is idempotent Thus calling this method multiple times will always update the same resource multiple times.
* [HEAD](https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them)
* [DELETE](https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them)
* [PATCH](https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them)
* PATCH: Submits a partial modification to a resource. If you only need to update one field for the resource, you may want to use the PATCH method.
* [OPTIONS](https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them)

PATCH vrs PUT

PUT method primarily fully replaces an entire existing resource but PATCH partially updates an existing resource.

The PATCH method is not a substitute to the PUT method. It applies a delta (diff) rather than replacing the entire resource.

* What is the difference between FromBody and FromUri?

Ans: The [FromUri] attribute is prefixed to the parameter to specify that the value should be read from the URI of the request, and the [FromBody] attribute is used to specify that the value should be read from the body of the request.

* Action Result and IAction Result in controller

Ans:

[https://www.c-sharpcorner.com/blogs/iactionresult-vs-actionresult#:~:text=IActionResult%20is%20an%20interface%20and,MVC%2C%20say%20an%20XML%20result.](https://www.c-sharpcorner.com/blogs/iactionresult-vs-actionresult%23:~:text=IActionResult%20is%20an%20interface%20and,MVC%2C%20say%20an%20XML%20result.)

* Various Ways to Pass Data From Controller to View in MVC

Ans:

<https://www.c-sharpcorner.com/UploadFile/abhikumarvatsa/various-ways-to-pass-data-from-controller-to-view-in-mvc/>

* Boxing and unboxing

Boxing converts a Value Type variable into a Reference Type variable

int num = 2020;

object obj = num;

num = 100;

Unboxing conversions

An unboxing conversion permits an explicit conversion from type object to any value-type or from any interface-type to any value-type that implements the interface-type

object box = 12;

int i = (int)box;

* C# compilation step

Ans:

<https://www.c-sharpcorner.com/UploadFile/a8024d/C-Sharp-program-compliation-steps/>

* Type of Inheritance

Ans:

<https://www.geeksforgeeks.org/c-sharp-inheritance/>

// single inheritance

class Animal {

public void Eat() {

Console.WriteLine("Animal is eating.");

}

}

class Dog : Animal {

public void Bark() {

Console.WriteLine("Dog is barking.");

}

}

<https://www.geeksforgeeks.org/c-sharp-multilevel-inheritance/>

<https://www.c-sharpcorner.com/blogs/multiple-inheritance-in-c-sharp-using-interfaces1>

* LINQ

Ans:

<https://www.c-sharpcorner.com/UploadFile/72d20e/concept-of-linq-with-C-Sharp/>

* Exception Handling in Asp.net
* Authentication and authorization

Ans: **authentication is the process of verifying who someone is, whereas authorization is the process of verifying what specific applications, files, and data a user has access to**

* == and === in javascript

Ans: == is used for comparison between two variables irrespective of the datatype of variable. === is used for comparision between two variables but this will check strict type, which means it will check datatype and compare two values

* Mapper

[https://www.simplilearn.com/tutorials/asp-dot-net-tutorial/automapper-in-c-sharp#:~:text=AutoMapper%20in%20C%23%20is%20a,maintains%20the%20conventions%20of%20AutoMapper](https://www.simplilearn.com/tutorials/asp-dot-net-tutorial/automapper-in-c-sharp%23:~:text=AutoMapper%20in%20C%23%20is%20a,maintains%20the%20conventions%20of%20AutoMapper).

* JWT

[https://www.akana.com/blog/what-is-jwt#:~:text=JWT%2C%20or%20JSON%20Web%20Token,after%20the%20token%20is%20issued.](https://www.akana.com/blog/what-is-jwt%23:~:text=JWT%2C%20or%20JSON%20Web%20Token,after%20the%20token%20is%20issued.)

Lazy loading static post put identity routing role .net core and asp net extension method

* Lazy and eager loading in EF

<https://www.c-sharpcorner.com/article/eager-loading-lazy-loading-and-explicit-loading-in-entity-framework/>

* Static class

[https://www.c-sharpcorner.com/UploadFile/74ce7b/static-class-in-C-Sharp/#:~:text=A%20static%20class%20in%20C%23,class%20that%20cannot%20be%20instantiated.](https://www.c-sharpcorner.com/UploadFile/74ce7b/static-class-in-C-Sharp/%23:~:text=A%20static%20class%20in%20C%23,class%20that%20cannot%20be%20instantiated.)

* Which is more secure post or get

**GET is less secure compared to POST** because data sent is part of the URL. So it's saved in browser history and server logs in plaintext. POST is a little safer than GET because the parameters are not stored in browser history or in web server logs

* Identity

<https://www.tutorialspoint.com/asp.net_core/asp.net_core_identity_overview.htm>

* Routing in controller

<https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-6.0>

* Managed and unmanaged in .net

<https://www.c-sharpcorner.com/uploadfile/puranindia/managed-code-and-unmanaged-code-in-net/>

* Role in controller

[https://codewithmukesh.com/blog/permission-based-authorization-in-aspnet-core/#:~:text=Role%2DBased%20Authorization%20in%20ASP,on%20his%2Fher%20role%20settings.](https://codewithmukesh.com/blog/permission-based-authorization-in-aspnet-core/%23:~:text=Role-Based%20Authorization%20in%20ASP,on%20his%2Fher%20role%20settings.)

* Middleware

<https://www.c-sharpcorner.com/article/overview-of-middleware-in-asp-net-core/>

* Entity Framework Advantages and disadvantages

<https://cybarlab.com/advantages-and-disadvantages-of-ef>

* Soap and rest api

[https://www.geeksforgeeks.org/difference-between-rest-api-and-soap-api/#:~:text=SOAP%20uses%20only%20XML%20for,but%20SOAP%20cannot%20use%20REST.](https://www.geeksforgeeks.org/difference-between-rest-api-and-soap-api/%23:~:text=SOAP%20uses%20only%20XML%20for,but%20SOAP%20cannot%20use%20REST.)

* IEnumerable and IQueryable
* Object pooling
* Mediator
* DbContext class
* Model builder in entityframework core
* Difference between ToList, AsEnumerable, AsQueryable
* Why model builder call static function
* Detaile study of static class, virtual keyword
* **What is polymorphism**

Polymorphism is a fundamental concept in object-oriented programming (OOP) that allows objects of different classes to be treated as objects of a common base class. It enables a single interface to be used to represent different types of objects, promoting flexibility and extensibility in software design.

There are two main types of polymorphism:

Compile-time Polymorphism (Method Overloading): This type of polymorphism occurs when there are multiple methods in a class with the same name but different parameter lists (different types or different numbers of parameters). The appropriate method to be called is determined at compile time based on the number and types of arguments provided.

Run-time Polymorphism (Method Overriding): Run-time polymorphism occurs when a subclass provides a specific implementation of a method that is already defined in its superclass. This is achieved through method overriding. The decision about which implementation of the method to call is made at run time, based on the actual type of the object.

Polymorphism is a key concept in achieving abstraction and encapsulation, two other principles of object-oriented programming. It allows you to write more generic and reusable code by focusing on the common behaviors and characteristics of related objects, rather than their specific implementations.

Here's a simple example in C# to illustrate run-time polymorphism through method overriding:

csharp

Copy code

class Shape

{

public virtual void Draw()

{

Console.WriteLine("Drawing a shape.");

}

}

class Circle : Shape

{

public override void Draw()

{

Console.WriteLine("Drawing a circle.");

}

}

class Square : Shape

{

public override void Draw()

{

Console.WriteLine("Drawing a square.");

}

}

class Program

{

static void Main(string[] args)

{

Shape shape1 = new Circle();

Shape shape2 = new Square();

shape1.Draw(); // Calls Circle's Draw method

shape2.Draw(); // Calls Square's Draw method

}

}

* Remove duplicate from string or list
* Creare Singelton Class

**Create Procedure for extract list with pagination**

CREATE PROCEDURE GetPaginatedList

@PageNumber INT,

@PageSize INT

AS

BEGIN

-- Calculate the starting index for the current page

DECLARE @StartIndex INT = (@PageNumber - 1) \* @PageSize

-- Select the data for the current page using OFFSET-FETCH

SELECT Column1, Column2, Column3

FROM YourTable

ORDER BY Column1 -- You can specify your own ordering column

OFFSET @StartIndex ROWS FETCH NEXT @PageSize ROWS ONLY;

END

Use Join IN LInq and lamda expression