**INTERVIEW QUESTIONS**

C#:

OOPs and C#:

1. Four pillars of OOPs
2. Abstract class and abstraction concepts- Is it same or different?
3. Interface
4. Inheritance
5. Value type and Reference type
6. How to achieve abstraction without interface & abstract class?

namespace AbstractionWithoutInterfaceAndAbstractClass

{

class Employee

{

public int EmpId;

public string EmpName;

public double GrossSalary;

double taxDeduction = 0.1;

double netSalary;

public Employee(int EId, string EName, double EGrossPay)

{

this.EmpId = EId;

this.EmpName = EName;

this.GrossSalary = EGrossPay;

}

//abstraction achieve because of private method we can hide the implementation

private void CalculateSalary()

{

if (GrossSalary > 30000)

{

netSalary = GrossSalary - (taxDeduction \* GrossSalary);

Console.WriteLine("Salary is : {0}",netSalary);

}

else

{

Console.WriteLine("Salary is : {0}",GrossSalary);

}

}

public void ShowEmployeeDetails()

{

this.CalculateSalary();

}

}

class Program

{

static void Main(string[] args)

{

Employee e = new Employee(1,"Keshav Kumar",35000);

e.ShowEmployeeDetails();

}

}}

1. Abstract class object can be created or not? If no, then why?

* No, we can’t create an instance of an abstract class.
* Because an abstract class is an incomplete class (incomplete in the sense it contains abstract methods without body and output.)

1. Copy Constructor?

* A constructor that creates an object by copying variables from another object or that copies the data of one object into another object is termed as the **Copy Constructor**.
* It is a parameterized constructor that contains a parameter of the same class type.
* The main use of copy constructor is to initialize a new instance to the values of an existing instance.
* Normally, C# does not provide a copy constructor for objects, but if you want to create a copy  [constructor](https://www.geeksforgeeks.org/c-sharp-constructors/)in your program you can create according to your requirement.

Code:

namespace ConstructorCopy

{

public class Program

{

string name;

public Program(string name1)

{

name = name1;

}

//copy constructor

public Program(Program p)

{

name = p.name;

}

static void Main(string[] args)

{

Program p1 = new Program("Keshav");

Console.WriteLine("Name of p1 :"+p1.name);

//copy constructor

Program p2 = new Program(p1);

Console.WriteLine("Name of p2 :"+p2.name);

}

}

}

1. What is Constructor Chaining?

* constructor chaining*,* where a constructor calls another constructor in its class using the ": this ()

The key is the: **this ()** keyword, which is a reference to another constructor. The order of the call is as follows: First, we call the constructor which is referenced with the: **this ()** keyword, and if that also references another constructor, it will also call that constructor, climbing up the call chain.

Example:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ArrayReverseDemo

{

class ConstructorChainningDemo

{

public ConstructorChainningDemo(): this(10)

{

Console.WriteLine("First Constructor is calling");

}

public ConstructorChainningDemo(int Age)

{

Console.WriteLine("Second Constructor is Calling and age is "+Age);

}

static void Main()

{

ConstructorChainningDemo constructorChainningDemo = new ConstructorChainningDemo();

}

}

}

1. Can you create a function in C# which can accept varying number of arguments?

* **The params keyword lets you specify a method parameter that takes a variable number of arguments.**

1. What happens if static class can have constructor?

A static constructor is used to initialize any static data, or to perform a particular action that needs performed once only. It is called automatically before the first instance is created or any static members are referenced.

Static constructors have the following properties:

* A static constructor does not take access modifiers or have parameters.
* A static constructor is called automatically to initialize the class before the first instance is created or any static members are referenced.
* A static constructor cannot be called directly.
* The user has no control on when the static constructor is executed in the program.
* A typical use of static constructors is when the class is using a log file and the constructor is used to write entries to this file.
* Static constructors are also useful when creating wrapper classes for unmanaged code, when the constructor can call the LoadLibrary method.

1. Parallelism or parallel foreach?
2. Difference between var and dynamic?

* Difference between Var and Dynamic is:

|  |  |
| --- | --- |
| VAR | DYNAMIC |
| It is introduced in C# 3.0. | It is introduced in C# 4.0. |
| The variables are declared using var keyword are statically typed. | The variables are declared using dynamic keyword are dynamically typed. |
| The type of the variable is decided by the compiler at compile time. | The type of the variable is decided by the compiler at run time. |
| The variable of this type should be initialized at the time of declaration. So that the compiler will decide the type of the variable according to the value it initialized. | The variable of this type need not be initialized at the time of declaration. Because the compiler does not know the type of the variable at compile time. |
| If the variable does not initialize it throw an error | If the variable does not initialize it will not throw an error. |
| It cannot be used for properties or returning values from the function. It can only used as a local variable in function. | It can be used for properties or returning values from the function. |
| var myvalue = 10; // statement 1 myvalue = “GeeksforGeeks”; // statement 2 Here the compiler will throw an error because the compiler has already decided the type of the myvalue variable using statement 1 that is an integer type. When you try to assign a string to myvalue variable, then the compiler will give an error because it is violating safety rule type. | dynamic myvalue = 10; // statement 1 myvalue = “GeeksforGeeks”; // statement 2 Here, the compiler will not throw an error though the type of the myvalue is an integer. When you assign a string to myvalue it recreates the type of the myvalue and accepts string without any error. |

1. Exceptional Handling
2. Can we have return statement inside finally block?

* No

1. Can we have return statement inside catch block?

* Yes

1. Access Modifiers in C#?
2. What is Abstract and Interface? What is the difference between them? What is similarity between them?
3. Explain Dependency Injection? How can we achieve it?
4. Async and await in C#?

* Async and await keywords of C# were introduced in C# 5.0. They were designed to make it easier to write asynchronous code, which can run in the background while other code is executing.
* The "async" keyword marks a method asynchronous, meaning it can be run in the background while another code executes. When you mark a method as async, you can use the "await" keyword to indicate that the method should wait for the result of an asynchronous operation before continuing.

1. What is Static classes?
2. Can this be used within a static method?

* We can not use this in static method because keyword this returns a reference to the current instance of the class containing it.
* Static methods or any static member do not belong to a particular instance. They exist without creating an instance of the class and call with the name of a class not by instance of the class and call with the name of class not by instance so we can not use this keyword in the body of the static Methods, but in case of Extension Methods, we can use it as the function’s parameters.

1. Why doesn’t C# allow static methods to implement an interface?

* Static classes cannot be inherited whereas an interface must be inherited to use it. Making an interface static, would render it pretty much useless. You'd need to inherit it in a class to implement it's members but static would prevent you from doing exactly that

1. Exception Handling in C#? How can we use?
2. What is is/as/ref/out keyword? explain each?

The difference between [*is*](https://www.geeksforgeeks.org/c-is-operator-keyword/) and [*as*](https://www.geeksforgeeks.org/c-as-operator-keyword/) operators are as follows:

* The **is** operator is used to check if the run-time type of an object is compatible with the given type or not whereas **as** operator is used to perform conversion between compatible reference types or Nullable types.
* The **is** operator is of Boolean type whereas **as** operator is not of Boolean type.
* The **is** operator returns true if the given object is of the same type whereas **as** operator returns the object when they are compatible with the given type.
* The **is** operator returns false if the given object is not of the same type whereas **as** operator return null if the conversion is not possible.
* The **is** operator is used for only reference, boxing, and unboxing conversions whereas **as** operator is used only for nullable, reference and boxing conversions.

1. What is Partial Class?
2. What is Sealed Class?
3. What is Static keyword?
4. What is the use of Yield keyword in C#?

* The Yield keyword will act as an iterator blocker and generate or return values.
* "The yield keyword signals to the compiler that the method in which it appears is an iterator block. The compiler generates a class to implement the behavior that is expressed in the iterator block. In the iterator block, the yield keyword is used together with the return keyword to provide a value to the enumerator object. This is the value that is returned, for example, in each loop of a foreach statement. The yield keyword is also used with a break to signal the end of the iteration."

Code:

Class Program

{

Static IEnumerable<int> Y()

{

yield return 1;

yield return 2;

yield return 3;

}

Static void Main(string[] args)

{

foreach(int I in Y())

{

Console.WriteLine(i);

}

}

}

Output:1,2,3

1. What is managed and unmanaged code?

Managed Code and Unmanaged Code

**Managed code** is the code which is managed by the CLR (Common Language Runtime) in *.NET Framework*. Whereas the Unmanaged code is the code which is directly executed by the operating system. Below are some important differences between the Managed code and Unmanaged code:

| **Managed Code** | **Unmanaged Code** |
| --- | --- |
| It is executed by managed runtime environment or managed by the CLR. | It is executed directly by the operating system. |
| It provides security to the application written in .NET Framework. | It does not provide any security to the application. |
| It provides runtime services like Garbage Collection, exception handling, etc. | It does not provide runtime services like Garbage Collection, exception handling, etc. |
| The source code is compiled in the intermediate language known as *IL or MSIL or CIL*. | The source code directly compiles into native languages. |
| Memory is managed by CLR’s Garbage Collector. | Memory is managed by the programmer. |
| Performance is Slightly slower due to the overhead of memory management and JIT compilation. | Performance is Faster due to direct access to system resources and compiled machine-specific code. |

1. If we have multiple catch block and exception occurs than what happen? Will all catch block executed?

At a time only one Exception is occurred and at a time only one catch block is executed.

All catch blocks must be ordered from most specific to most general.

Code:

namespace MultipleCatchBlock

{

internal class Program

{

static void Main(string[] args)

{

try

{

int a = 10;

int b = 0;

int c = a / b;

Console.WriteLine(c);

string f = null;

Console.WriteLine(f.Length);

int[] arr = new int[3];

arr[0] = 11;

arr[1] = 12;

arr[2] = 13;

arr[3] = 14;

foreach (var item in arr)

{

Console.WriteLine(item);

}

}

catch (DivideByZeroException ex)

{

Console.WriteLine(ex.Message);

}

catch (NullReferenceException ex)

{

Console.WriteLine(ex.Message);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

}

1. What is Garbage Collection? Can we use garbage collection forcefully in our code?
2. What is difference between throw and throw ex?
3. Can we use async keyword without await keyword?

Certainly! The async keyword in C# is used to indicate that a method contains asynchronous code. However, it’s important to understand that the async keyword itself doesn’t inherently make a method asynchronous. Instead, it allows you to use the await keyword within the method to perform asynchronous operations.

Here are a few points to consider:

1. **Method Execution Behavior**:
   * When you mark a method as async, it doesn’t automatically run asynchronously. The method still executes synchronously until an await expression is encountered.
   * [If there’s no await inside the method, it behaves like a regular synchronous method1](https://stackoverflow.com/questions/47938520/what-exactly-happens-when-you-call-an-async-method-without-the-await-keyword).
2. **Using**async**Without**await:
   * You can technically use the async keyword without any await expressions, but it’s generally not recommended.
   * If you don’t use await, the method will execute synchronously, and the benefits of asynchronicity won’t be realized.
   * [In most cases, you should avoid creating async methods without any await calls2](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/async).
3. **Suppressing Warnings**:
   * When you call an async method without awaiting it, you’ll receive a warning (e.g., “Because this call is not awaited, the current method continues to run before the call is completed”).
   * If you’re sure that you don’t want to wait for the asynchronous call to complete and that the called method won’t raise any exceptions, you can suppress the warning.
   * [However, be cautious about exceptions that might occur within the method3](https://stackoverflow.com/questions/15522900/how-to-safely-call-an-async-method-in-c-sharp-without-await).
4. What is Reflection?
5. What is the difference between Equality Operator (==) and Equals () method in C#?
6. How can we prevent a class from overriding in C#? use sealed keyword.
7. Why is the virtual keyword used in code?

* The virtual keyword is used while defining a class to specify that the methods and the properties of that class can be overridden in derived classes.

1. What is the difference between a Struct and a Class in C#?

* Class and struct both are the user defined type but have some major differences:

STRUCT

* The Struct is value type in C# and it inherits from System.Value type.
* Struct is usually used for smaller amounts of data.
* Struct can’t be inherited to other type.
* A Struct can’t be abstract.

Class

* The Class is reference type in C#, and it inherits from the System.Object type.
* Classes are usually used for larger amounts of data.
* Classes can be inherited to other class.
* A class can be abstract type.
* We can create a default constructor.

1. What are the different types of classes in C#?

There are 4 types of Classes in C#.

* Partial Class: Allows its members to be divided or shared with multiple .cs files. It is denoted by the keyword Partial.
* Sealed Class: It is a class which cannot be inherited. To access the members of sealed class, we need to create the object of the class. It is denoted by keyword Sealed.
* Abstract Class: It is a class whose object cannot be instantiated. The class can only be inherited. It should contain at least one method. It is denoted by the keyword Abstract.
* Static Class: It is a class which does not allow inheritance. The members of the class are also static. It is denoted by the keyword static. This keyword tells the compiler to check for any accidental instances of the static class. We can not create instances of the static class.

1. Can we give implementation of method in Interface in C# 8.0?

* Before C# 8.0 interfaces only contain the declaration of the members (methods, properties, events and indexers), but from C# 8.0 it is allowed to add members as well as their implementation to the interface. Now we are allowed to add a method with their implementation to the interface without breaking the existing implementation of the interface, such type of methods is known as default interface methods (also known as the virtual extension methods).

1. What is difference between == vs equals?

* The Equality Operator ( ==) is the comparison operator and the Equals() method in C# is used to compare the content of a string.
* In C#, the [equality operator ==](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/equality-operators) checks whether two operands are equal or not, and the [Object.Equals()](https://docs.microsoft.com/en-us/dotnet/api/system.object.equals?view=net-6.0#system-object-equals(system-object)) method checks whether the two object instances are equal or not.

Example1:

int i = 10, j = 10;

Console.WriteLine(i == j); // true

Console.WriteLine(i.Equals(j)); // true

Example2:

string str1 = "Hello",

str2 = "Hello",

str3 = "hello";

Console.WriteLine(str1 == str2); // true

Console.WriteLine(str1 == str3 ); // false

Console.WriteLine(str1.Equals(str2));// true

Console.WriteLine(str1.Equals(str3));// false

1. Why C# not supported Multiple Inheritance? And what is Dimond problem?
2. What is difference between the Encapsulation and Abstraction?
3. What are the new features of Interface after C# 8?

* An Interface contains methods with default implementation or with body.
* Interface can have static fields but not instance fields because we cannot create instance/ object of an interface.
* In C#, by default, all members declared in an interface have public as the access modifier. But now they allow explicit access modifiers.
* We can do explicit interface implementation in interface.
* Now, we can create in Interface:
* Methods
* Properties
* Indexers
* Events
* Constants
* Static Constructors
* Static Fields
* Access Modifiers (default is Public).
* Default implementation of methods.
* Operators
* Nested Types

LINQ:

1. What is LINQ? Why we use LINQ if we have SQL Server?
2. Difference between First and FirstOrDefault?
3. Difference between Single and SingleOrDefault?
4. What is difference between IQueryable vs IEnumerable?
5. How IEnumerable works?
6. Write a simple LINQ query.
7. What is Group By in LINQ and how we use it write some code for that?

Database:

1. What are Transactions?
2. What is Store Procedure? How can we create Store Procedure?
3. What is clustered and unclustered Index?
4. Db Connection handling?
5. Aggregate functions in SQL Server?
6. What is Joins? Explain each type of Joins?
7. What is Group by?
8. What is Indexing?
9. What is Rank and Dense Rank?
10. What is Function in SQL Server?
11. What is the difference between Store Procedure and Function?

Difference between functions and stored procedures in SQL Server:

|  |  |
| --- | --- |
| FUNCTIONS | STORED PROCEDURES |
| Function must return a value. | Stored Procedure may or may not return values. |
| Create function fn\_Addtion\_Of\_Number(@int1 as int)  Returns int  As  begin  return (@int1+@int1)  end | Create procedure spGetData  as  begin  select \* from student  end |
| Function can have only input parameters. | Stored Procedures can have both input or output parameters. |
| Functions can be called from Stored Procedures. | Stored Procedures cannot be called from a Function. |
| Create function fn\_Addtion\_Of\_Number(@int1 as int)  Returns int  As  begin  return (@int1+@int1)  end | Create procedure spAddition  As begin  Selectv dbo.fn\_Addition\_Of\_Numbers(5)  End  Execute spGetAddition |
| Functions allows only SELECT statement in it. | The Stored Procedure allows SELECT as well as DML (INSERT/UPDATE/DELETE) statement in it. |
| Function can be embedded in a select statement. | Stored Procedures cannot be utilized in a SELECT statement. |
| Functions can be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section. | Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section. |
| Try-catch block cannot be used in a Function. | An exception can be handled by try-catch block in a Stored Procedure. |
| We cannot use Transactions in Function. (Transactions: Commit, Rollback concepts) | We can use Transactions in Stored Procedure. |
| We use SELECT command to execute a function. | We use EXEC or EXECUTE keywords to execute a stored procedure. |
| Functions are computed values and cannot perform permanent environmental changes to SQL Server (i.e., no INSERT or UPDATE statement allowed). | Stored Procedures can perform permanent environmental changes to SQL Server (i.e. INSERT or UPDATE statement are allowed). |
| A Function can be used in join clause as a result set. | Procedures cannot be used in Join clause. |

1. What is View and how can we create?
2. Why we use View?
3. What is triggers and how can we create and use?
4. What are the different types of Triggers in SQL Server?
5. Find Nth highest salary from Employee table?

select min(Salary) from(

select top N Salary as Salary from Employee\_Details

order by Salary desc)

as Salary

1. Find duplicate Employee from Employee table?

select E.Name,Count(\*) from Employee\_Details as E

group by Name having Count(\*)>1;

1. Delete duplicate Employee from Employee table?

delete from Employee\_Details

where id not in(

select min(id) from Employee\_Details

group by Name

)

1. What is the meaning of the:

Select 1 from table.

if exists (select 1 from Employee\_Details where Salary>43000)

begin

print'There are employees with salary greater than 43000'

end

else

begin

print 'There is no employee with salary greater than 43000 '

end

1. What is the output of Select 1+1;?
2. Can we use Limit in SQL Server?

No

Question on Stored Procedure:

1. What is Stored Procedure?

* A stored procedure is a database object.
* A stored procedure is a series of declarative SQL statements.
* A stored procedure can be stored in the database and can be reused over and over again.
* Parameters can be passed to a stored procedure, so that the stored procedure can act based on the parameter value(s).
* SQL Server creates an execution plan and stores it in the cache.
* Why use stored procedure?
* Suppose hamare pass ek query hai jisko hume hmesha use krna hai to wske liye hm ek stored procedure bnayege jiske andar wo query likh dege aur jab bhi hume ws query ka use prega hm iss SP ko call kar lege. Jab hm procedure create krte hain to ye SQL Server system ke cache mai save ho jaata. Pehli baar ye create krega ws time ye cache mai save ho jaayega aur phir next time jab hm isko call krege to ye cache se value return kr dega isse query fast ho jaata hai.

1. Types of Stored Procedures?

* User Defined Stored Procedure
* User defined stored procedures are created by database developers or database administrators. These SPs contains one or more SQL statements to select, update, or delete records from database tables.
* System Stored Procedure
* System stored procedures are created and executed by SQL Server for the server administrative activities.

1. How to create stored procedure?

--create procedure for returning the details of Employee

create procedure spGetEmpDetails

as

begin

select \* from EmployeeDetails;

end

--calling procedure

spGetEmpDetails;

execute spGetEmpDetails;

exec spGetEmpDetails;

1. How to modify stored procedure?

* We can modify SP using ALTER keyword.

-- modify existing procedure

alter procedure spGetEmpDetails

as

begin

select \* from EmployeeDetails;

select \* from EmployeeDetails where City='Gurgaun';

end

--call procedure

exec spGetEmpDetails;

1. How to drop stored procedure?

* We can use DROP keyword for dropping SP.
* drop procedure spGetEmpDetails;

1. How to send parameter in stored procedure?

--input parameter

create procedure spGetEmpDetails

@id int,

@Name varchar(50)

as

begin

select \* from EmployeeDetails where id=@id;

select \* from EmployeeDetails where Name=@Name;

end

--calling procedure with parameter, sequance is matter

execute spGetEmpDetails 1,'Madhav'

--default input value

alter procedure spGetEmpDetails

@id int=1,

@Name varchar(50)='Madhav'

as

begin

select \* from EmployeeDetails where id=@id;

select \* from EmployeeDetails where Name=@Name;

end

--calling procedure if we call procedure without giving the parameter it will execute with default value

--and if we call procedure with parameter then it will give the priority of this parameter not default parameter

exec spGetEmpDetails

exec spGetEmpDetails 2,'Keshav'

--output parameter

create procedure spAddDigit

@Num1 int,

@Num2 int,

@Result int output

as

begin

set @Result=@Num1+@Num2;

end

--calling procedure with input,output parameter

declare @EId int

exec spAddDigit 2,4,@EId output;

select @EId;

1. Types of parameters in Stored Procedure?
2. How to encrypt stored procedure?

WEB API

1. What is Authentication & Authorization? How can we achieve it?
2. What is HTTP status code? Different- different status code meaning?
3. What are data validations?
4. HTTP Verbs?
5. Request Life Cycle?
6. What is middleware? How can we achieve it?
7. How middleware works? How to create custom middleware and how we call that middleware? Does middleware order matters?
8. What is Route?
9. What is the type of Route? Explain each?
10. How the APIs work?
11. How the request response workflow works?
12. Suppose we have three endpoints, and we want to use these endpoints in single API how to do that? Write code for that?
13. Explain request and response pipeline?
14. Can we create model validation in web api?
15. What is the difference between MVC Controller and Web API Controller in ASP.NET Core?

MVC is used for Web application development. MVC returns both data and view of the data whereas, Web API returns only the HTTP services data i.e. only in form of data. NOTE : Web APIs are the services for any type of devices and any clients but MVC provides service only to its clients.

1. What is the need of Web API if we can achieve it by using MVC to return JSON data?

Web API can return data as JSON, XML, and other formats, but MVC only returns data as JSON using JSONResult. MVC does not support content negotiation or self-hosting, while Web API does. Even though Web API supports features of MVC, like routing and model binding, they are different, coming from System.

Entity Framework:

1. What is Entity Framework?
2. What are the approaches of Entity Framework?
3. How can we achieve Code First Approach?
4. How can we achieve Database First Approach?
5. How can Migration have done?
6. If we update our model class, then how can update our database using code first?
7. How to handle transactions?
8. Why do we need to handle models if we already have an entity?

MVC:

1. What is MVC?
2. What are Sessions in MVC applications? How can we achieve it?
3. Responsibility of Model, Controller and View?
4. What is Action Method? What are the rules for creating Action Method?
5. What is IActionResult?
6. What is View?
7. What is Razor?
8. What is Razor View Engine?
9. Rules for creating Razor View?
10. What is Layout View (\_Layout.cshtml)?
11. What is View Start File? Why we use this file? ( \_ViewStart.cshtml)
12. What is \_Viewimports.cshtml file?
13. How can we pass data Controller to view?
14. What is ViewData/ViewBag/TempData?
15. What is Repository Pattern?
16. What is Tag Halper? How can we achieve it? What are the different type of Tag Happer in .net core?
17. What is Model Binding? How Model Binding works?
18. What is Model Validation?
19. Validation Tag Helper?
20. Built-in Tag Halper?
21. What are Filters? What are the different types of filters?

Filters are used to add cross-cutting concerns, such as Logging, Authentication, Authorization, Exception Handling, Caching, etc., to our application. In ASP.NET Core Applications, Filters allow us to execute cross-cutting logic in the following ways:

* Before an HTTP request is handled by a controller action method.
* After an HTTP request is handled by a controller action method.
* After the response is generated but before it is sent to the client.

Types of Filters are:

1. Authorization Filters

The Authorization Filter is used to perform Authentication and Authorization checks before an action method is executed.

Attributes:

* [Authorize]
* [AllowAnonymous]
* **Custom Authentication:** You can also create Custom Authentication. To do so, we need to create a class implementing the **IAuthorizationFilter** interface and provide implementations for the **OnAuthorization** method, where we need to write the custom authentication logic according to our business requirements.

1. Action Filters

The Action Filters in the ASP.NET Core MVC Application are executed before and after an action method is executed. They perform tasks like Logging, Modifying the Action’s Arguments, or Altering the Action’s Result.

1. Result Filters

The Result Filters in ASP.NET Core MVC Application runs after the action method has been executed but before the result is processed and sent to the client. This means we can modify the view or the result data before it gets rendered to the output stream. They are used for tasks such as Adding Headers to the response, Modifying the Result, etc.

1. Exception Filters

The Exception Filters are executed when an unhandled exception occurs during the execution of an action method. They are used for Logging, Error Handling, and Displaying Different Error Pages Based on the Error Status Codes.

1. What is the difference between Filters and Middleware?

* Scope of Application (Where they Applied):
* **Filters:** Filters are primarily applied to individual controller actions or controllers. They are used to add specific behaviors or concerns to processing a single action or a group of actions within a controller.
* **Middlewares:** Middleware are applied to the entire application’s request processing pipeline. They can handle requests and responses globally, regardless of the specific controller or action being invoked.

Execution Point (When they Execute):

* **Filters:** Filters execute within the ASP.NET Core Framework’s pipeline and are part of the controller/action execution process. They are triggered before or after the execution of a specific action method.
* **Middlewares:** Middlewares execute earlier in the Request Processing Pipeline, typically before the request reaches the ASP.NET Core MVC Controller action method. They can intercept requests and responses and perform tasks at various stages of the request processing pipeline, such as Routing, Authentication, Response Formatting, etc.

**Purpose and Concerns (Why they are Used):**

* **Filters:** Filters are designed to handle concerns specific to the ASP.NET Core MVC framework, such as Logging, Authentication, Authorization, Exception Handling, Caching, Custom Logic, etc.
* **Middlewares:** Middlewares are more general-purpose and can handle many concerns, including Routing, Authentication, Request/Response Logging, Compression, Security, etc.

**Configuration (How they Configured):**

* **Filters:** Filters are typically configured using attributes (e.g., **[Authorize], [AllowAnonymous], etc.**) on controllers or action methods. You can also register global filters in the Program Class.
* **Middlewares:** Middlewares are configured and ordered in the Program class (e.g., **UseHttpsRedirection(), UseAuthorization(), MapControllers(), etc**).

**Execution Order (What is the Execution Order):**

* **Filters:** The execution order for filters is decided based on the type of filters you are applying to the controllers and action methods. So, the order of Filters is not important.
* **Middlewares:** The execution order for middleware components is determined by the order in which they are added to the IApplicationBuilder pipeline, i.e., to the Request Processing Pipeline. So, the order of Middleware Components is important.

1. What is Routing?
2. How can we register filters in MVC application?
3. What will happen if we not use IActionResult and use some othe datatype for return type?

.NET CORE BASICS CONCEPTS:

1. Middleware?
2. Singletone vs scoped vs transient?
3. Program.cs class?
4. Data Annotations?
5. Partial views?
6. Dependency Injection?
7. Difference between AddSingleton, AddScoped, AddTransient?

DESIGN PATTERNS:

1. What is the design pattern? How many design patterns you used?
2. Which design patterns you used in your project? How to implement that design pattern?
3. Explain Repository Pattern?

Solid Principle:

1. What is SOLID Principles? What is the difference between SOLID Principles and Design Patterns?

SOLID Principle:

* SOLID Principles are a set of principles, which must be followed to develop flexible, maintainable, and scalable software systems.
* SOLID Principles aren’t concrete- rather abstract.

Design Pattern:

* Design patterns are concrete and solve a particular kind of problem in software’s.
* Design patterns are concrete and solve a particular kind of problem in a specific and fixed manner.

Q6 C#: How to achieve abstraction without interface & abstract class?

namespace AbstractionWithoutInterfaceAndAbstractClass

{

class Employee

{

public int EmpId;

public string EmpName;

public double GrossSalary;

double taxDeduction = 0.1;

double netSalary;

public Employee(int EId, string EName, double EGrossPay)

{

this.EmpId = EId;

this.EmpName = EName;

this.GrossSalary = EGrossPay;

}

//abstraction achieve because of private method we can hide the implementation

private void CalculateSalary()

{

if (GrossSalary > 30000)

{

netSalary = GrossSalary - (taxDeduction \* GrossSalary);

Console.WriteLine("Salary is : {0}",netSalary);

}

else

{

Console.WriteLine("Salary is : {0}",GrossSalary);

}

}

public void ShowEmployeeDetails()

{

this.CalculateSalary();

}

}

class Program

{

static void Main(string[] args)

{

Employee e = new Employee(1,"Keshav Kumar",35000);

e.ShowEmployeeDetails();

}

}}

Q C#: Can we have return statement inside finally block?

The finally block is a powerful construct used for cleanup operations. It executes regardless of whether an exception occurs or not. However, there are some rules and restrictions associated with it:

1. **Purpose of**finally**Blocks**:
   * The primary purpose of a finally block is to ensure that certain code runs no matter what happens in the preceding try or catch blocks.
   * It’s commonly used for resource management (e.g., closing files, releasing database connections, etc.).
2. **What You Can’t Do in a**finally**Block**:
   * **No**return**Statements**: You **cannot** have a return statement directly within a finally block.
   * **No**break**or**continue**Statements**: Similarly, you cannot use break or continue to exit a loop from within a finally block.
   * **No Jumping Out**: If you use goto, break, or continue within a finally block, the target label must be within the same finally block. [Otherwise, you’ll encounter a compile-time error1](https://stackoverflow.com/questions/36622877/jumping-out-of-a-finally-block).
3. **Why No**return**Statements in**finally**Blocks?**
   * Imagine this scenario:

**C#**

try

{

// Some code

return "OK";

}

catch

{

return "NOK";

}

finally

{

return "Finally"; // Illegal!

}

* + The value to return is already determined before entering the finally block. In this case, the compiler has already decided whether to return “OK” or “NOK”.
  + Allowing a different return value from the finally block would be confusing and counterintuitive.

1. Difference between var and dynamic?

|  |  |
| --- | --- |
| VAR | DYNAMIC |
| It is introduced in C# 3.0. | It is introduced in C# 4.0. |
| The variables are declared using var keyword are statically typed. | The variables are declared using dynamic keyword are dynamically typed. |
| The type of the variable is decided by the compiler at compile time. | The type of the variable is decided by the compiler at run time. |
| The variable of this type should be initialized at the time of declaration. So that the compiler will decide the type of the variable according to the value it initialized. | The variable of this type need not be initialized at the time of declaration. Because the compiler does not know the type of the variable at compile time. |
| If the variable does not initialize it throw an error | If the variable does not initialize it will not throw an error. |
| It cannot be used for properties or returning values from the function. It can only used as a local variable in function. | It can be used for properties or returning values from the function. |
| var myvalue = 10; // statement 1 myvalue = “GeeksforGeeks”; // statement 2 Here the compiler will throw an error because the compiler has already decided the type of the myvalue variable using statement 1 that is an integer type. When you try to assign a string to myvalue variable, then the compiler will give an error because it is violating safety rule type. | dynamic myvalue = 10; // statement 1 myvalue = “GeeksforGeeks”; // statement 2 Here, the compiler will not throw an error though the type of the myvalue is an integer. When you assign a string to myvalue it recreates the type of the myvalue and accepts string without any error. |

Q: What is Copy Constructor?

* A constructor that creates an object by copying variables from another object or that copies the data of one object into another object is termed as the **Copy Constructor**.
* It is a parameterized constructor that contains a parameter of the same class type.
* The main use of copy constructor is to initialize a new instance to the values of an existing instance.
* Normally, C# does not provide a copy constructor for objects, but if you want to create a copy  [constructor](https://www.geeksforgeeks.org/c-sharp-constructors/)in your program you can create according to your requirement.

Code:

namespace ConstructorCopy

{

public class Program

{

string name;

public Program(string name1)

{

name = name1;

}

//copy constructor

public Program(Program p)

{

name = p.name;

}

static void Main(string[] args)

{

Program p1 = new Program("Keshav");

Console.WriteLine("Name of p1 :"+p1.name);

//copy constructor

Program p2 = new Program(p1);

Console.WriteLine("Name of p2 :"+p2.name);

}

}

}