1. CORS-
   * Cross origin resource sharing
   * Due to browser security it is not allowed to access code from different domains.
   * (can be different domain, different subdomain, schema- http, https or different port)
   * To solve this W3 Standard was introduced which allows a server to make cross domain calls.
2. Api controller v/s Controller-
   * The major difference is that API controller do not return views, they only return data.
   * The web API can also return data in various formats such as JSON, xml and other formats based on the accept header of the request but the controller only returns data in JSON format using JsonResult
3. What is dependency Injection and what problem does it solve
   * DI is a technique to achieve Inversion of control between classes and their dependencies.
   * Startup class-

services.AddSingleton<IRepository,Repository>();

* + If we create a new instance of the service directly into the controller, it tightly couples the controller with the service (Repository) used.
  + Later if the implementation of the interface changes, we need to instantiate the new service into the controller in place of the old services. This seems easy when it is used only at a single place. But in a typical real world application, the service may have multiple implementations and in such a case all the implementations will require change. This is not only tedious but also error prone.
  + That is where DI is useful.

1. Ways to register dependency injection
   * AddSingleton- An instance is created, when the service is first requested and that single instance is used by all http requests throughout the application
   * AddScoped- With scoped we get the same instance within the scope of a given http request but a new instance across different http requests
   * AddTransient- a new instance is provided every time an instance is requested
2. .csproj file-
   * Editing of project file doesn’t require unload like MVC. Direct option to edit
   * Simple and clean as compared to MVC
   * File and references are not included like MVC
   * Target framework : set of API we want to make available to our application (netcoreapp2.2)
   * AspNetCoreHostingModel : Inprocess/OutOfProcess
   * PackageReference: Nuget packages to be included
3. Program.cs-
   * Console apps usually have a Main() method
   * Asp.Net application initially starts as a console application
   * Main method configures Asp.Net core and starts it and at that point it becomes an Asp.Net core web application
4. Interface? And why do we use interface?
   * Interface is an abstract and a public class in which methods are declared
   * A class implementing the interface defines all the methods declared in an interface
   * The major reasons to use an interface is to:
     1. Make secure applications- because with interface only required information is exposed to the user
     2. Multiple inheritance- as we know multiple inheritance is not supported in c#, interfaces can help achieve that functionality as one class can inherent multiple interface
5. Difference between interface and abstract
   * An abstract class can define and declare methods which a class can implement or override whereas an interface can only define functionalities
   * A class can implement only one abstract class but a class can implement multiple interfaces and so multiple inheritance can be achieved by interface but not by abstract class
   * An abstract class can have a constructor but an interface does not have a constructor
6. State Management in Asp.Net
   * Since Asp.Net applications are stateless, we need state management techniques to preserve object values during multiple requests
   * So we can say that state management is used to store user information till the end of the user session
   * Server Side
     1. Session
     2. Application
     3. Cache
   * Client Side
     1. Cookies – Persistent, Non-persistent
     2. Hidden fields
     3. View State
     4. Query string
7. Difference between session and application state object
   * Both session and application are used to store and access variable from any page.
   * The major difference is that all users share one common application object whereas there is one session object for each user
8. TempData, ViewBag, ViewData
   * ViewBag and ViewData are used to take data from controller to view.
   * ViewBag is a dynamic property whereas ViewData is a dictionary object
   * ViewBag doesn’t require type casting or doesn’t perform null checks whereas ViewData requires type casting and performs null checks
   * In ViewBag data is passed as a property of ViewBag whereas in ViewData data is passed in key-value pair
   * TempData is used to take data from one controller to another or from one action to another
9. Delegates
   * A delegate is a type safe function pointer
   * A delegate is used when we want to pass a function as a parameter or when we have callback functions
10. Anonymous function
    * A function as the name suggests is a function that doesn’t have a name
    * Delegate keyword is used to create an anonymous method
    * It is not stored in a program file but is associated with a variable whose data type is function\_handle
11. Sealed class
    * Sealed classes are used to prevent inheritance (A class that is declared as sealed cannot be inherited by another class)
    * A method can be declared as sealed using the virtual keyword in the inherited class
    * A method declared as sealed cannot be overridden
12. Generic
    * A generic class as the name suggests means general form
    * Class DataStore<T>

{

Public T Data { get; set; }

}

* + DataStore<string> store=new DataStore<string>();
  + Store.Data=”Hello world”; //correct
  + Store.Data=1; //compile time error
  + A generic class increases reusability

1. Exceptions
   * An exception is a problem that arises in the execution of a program
   * There are various built in exceptions such as NullReferenceException, NumberNotFounException etc
   * Ways to handle exceptions
     1. Try-catch-finally blocks
     2. Manual exception thrown using ‘throw’ keyword
     3. Custom Exceptions created implementing the base ‘Exception’ class
2. Join in SQL
   * Inner Join- returns value that have matching records in both the tables
   * Left Join- returns all records from left table and matched records from right table
   * Right Join- returns all records from right table and matched records from left table
   * Full Join- returns all records when there is a match in either left or right table
   * Self-join- table joined with itself
3. Stored procedure and functions
   * SP is reusable code in database which is compiled once and the execution plan is stored. Every time a SP is called only the execution plan is invoked whereas in case of a function, a function is compiled every time it is invoked
   * SP can have input as well as output parameters whereas a function can only have input parameters and it performs the action and returns a result as a single value or a table
   * Functions must have a return value compulsorily whereas it is optional in case of a SP
   * SP can have DML, DQL whereas functions only have DQL(Data Query Language- Select)
   * Functions can be called from procedures but procedures cannot be called from functions
4. How does .Net framework work?
   * .Net code can be written in different languages like C#, F#, VB. These languages are compiled into Common Intermediate Language (CIL/MSIL) using the native compiler
   * Compiled code is stored in the form of an assembly file that has a .dll or .exe extension
   * When the code runs, the Common Language Runtime (CLR) converts the CIL code to the machine code using the Just In time compiler (JIT)
   * Then this machine code is executed on the specific architecture the computer is running on
5. Components of .Net framework
   * CLR
     1. Converts MSIL to machine code using JIT
     2. Services: memory management, garbage collection, exception handling, security, thread
   * CTS- Common Type System
     1. It is used to make sure that data defined in different languages should interact with each other to share information
   * CLS- Common Language Specification
     1. Subset of CTS and defines rules and regulations to be followed by every .Net framework language
6. Stack and heap memory
   * Stack-Used by local variables of a method. When a method is invoked a stack memory is allocated to and the stack is cleared when the method returns
   * Heap- All dynamic memory requirements of an application are fulfilled from the heap memory. After allocating some memory from heap to a variable, once its job is completed the memory must be returned back to heap so that the same can be used for another variable
   * Both Stack and Heap are virtual memories and reside on the RAM
7. Value Type VS reference type
   * A data type is a value type if it holds a data value within its own memory stack. – (int)
   * A reference type holds a reference to where the value is stored on the heap (string, object)
   * Value type can be on stack or heap depending upon the scope of the variable. But the value of a reference type always goes on a heap
8. Pillars of oop
   * Inheritance
   * Encapsulation
   * Abstraction
   * Polymorphism
9. What is encapsulation
   * Encapsulation can be referred as putting everything that is required to do a job into a capsule and presenting only the capsule to the user
   * So encapsulation is the process of binding data members and member functions of a program together to do a specific job, without revealing unnecessary details
10. What is polymorphism
    * Poly means many and morph means shape. Therefore polymorphism means something that has many shapes
    * Polymorphism refers to a process by which some code acts differently under different circumstances
    * There are two types of polymorphism:
      1. Compile time polymorphism- Here the compiler decides what shape is to be taken by an entity (example- overloading example and decided at compile time which method will be called)
      2. Runtime polymorphism- this polymorphism happens at runtime (example) class AnyVehicle{

Public void move(){

Console.log(“Any vehicle”);

}

}

Class Bike extends AnyVehicle{

Public void move(){

Console.log(“Bike”);

}

}

Public static void main(String[] args){

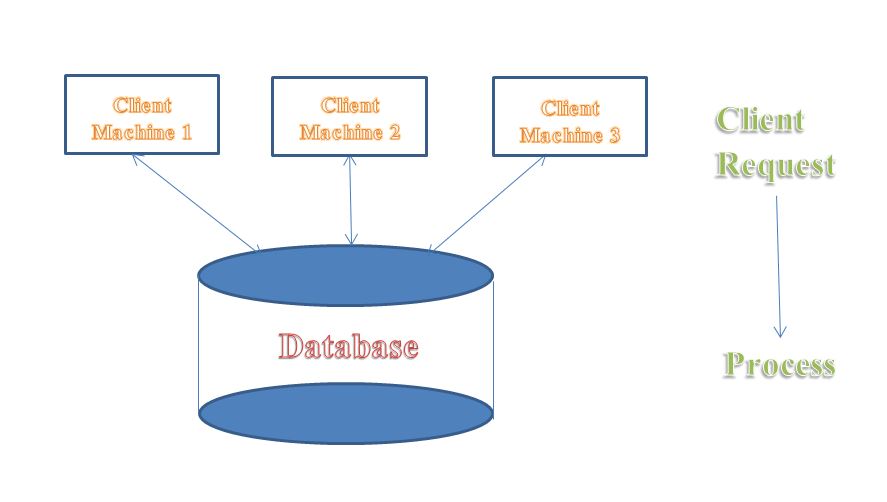
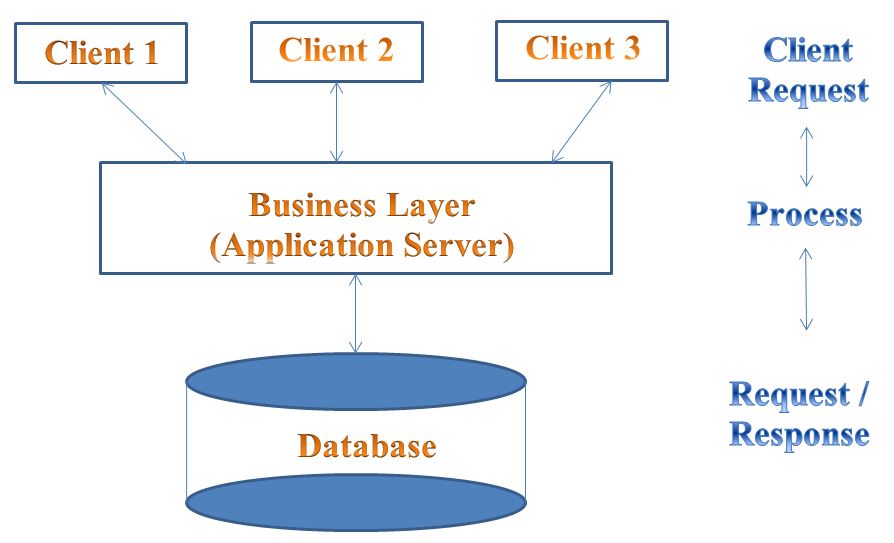
Anyvehicle vehicle=new Bike();

vehicle.move() //output- bike

vehicle=new AnyVehicle();

vehicle.move() //output- any vehicle

}

1. What is inheritance
   * Inheritance means receiving some qualities from a parent.
   * Inheritance is the mechanism by which a class (child) is created using definition of another class (parent)
   * It not only makes the implementation simple but also facilitates code reuse
2. What is abstraction
   * As a user if you have a problem, you don’t want to know how a component works or how is it made. You only want to know how it solves your problem
   * Abstraction is hiding unnecessary details from the necessary ones
3. Difference between abstraction and encapsulation
   * Abstraction solves the problem at design level whereas encapsulation solves the problem at a implementation level
4. Content negotiation in web API
   * When a client requests an API and if they want the response in a particular format, it can be specified in the Accept header and this is known as content negotiation
   * The user can also tell the web API about the request format they are sending using the content-type header
   * By default the response is returned in JSON format
   * Accept: application/xml
5. Authentication is MVC
   * No authentication
   * Individual user accounts
   * Work or school accounts
   * Windows authentication
6. Garbage collection
   * In .Net framework, automatic garbage collection is performed. Garbage collector is a component of CLR
   * Garbage collection works on the Managed heap
   * Conditions when GC is performed are
     1. If there is less physical memory, the GC is necessary
     2. If the memory used exceeds a pre-defined threshold, then GC occurs
     3. If GC.Collect method is called, this is done in unusual cases
   * System namespace is used
7. 2 tier architecture (DBMS)
   * It has 2 layers
   * Advantages:
     1. Maintenance- easy to maintain as only 2 layers
     2. Easy to understand
   * Disadvantages:
     1. Scalability- tough to scale because only a limited amount of clients can be handled. Many clients can make changes at the same moment
     2. Security- since direct connection with database, security is compromised
8. 3 tier architecture
   * 3 layer architecture
   * Advantages
     1. Scalability- can scale as many clients as required
     2. Security- security is Improved because no client directly interacts with the database
     3. High performance
     4. Light weight
   * Disadvantages
     1. Maintenance- tough to maintain
     2. Network traffic increased
     3. Increases complexity
9. What is temp table? Where is it stored in database?
   * Temporary tables are used when we want to store some intermediate or temporary result.
   * It is automatically deleted as soon as the last connection is terminated
   * Syntax:

Create table #EmployeeTb(id int, Name varchar(MAX))

* + Types:
    1. Local temporary table:

It is available only for the session it is created for. It is dropped when the connected that created it is closed.

Single “#” is used to create that table

If a temporary table is created during the execution of a SP, it is dropped automatically when the execution of SP is completed

* + 1. Global temporary table:

It is visible to all connections and dropped when last connection referencing to the table is closed

“##” is used to create a global temp table

1. What is memory table?
   * A memory table is a table which is used to store data in RAM and not on disk
   * To create a memory table we need to create in memory optimized file group and file in the database
   * A memory table is used to optimize transactions (performace benefit)
   * Syntax:

Create table memorytable(id int, name varchar(25))

with(memory\_optimized=on,durability=schema\_and\_data)

* + Durability can have “schema\_and\_data” or “schema”

1. What is trigger? Types of triggers?
   * A trigger is a special kind of stored procedure which is automatically fired/executed when a DDL or a DML statement related to the trigger is executed
   * Syntax:

Create trigger trigger\_name

(before | after)

[Insert | Update | Delete]

On table\_name

[For each row | For each column]

[Trigger\_body]

* + Types:
    1. DML Triggers:
       1. After triggers-

Fired after DML statement

* + - 1. Instead of triggers-

Fired instead of DML statement

* + 1. DDL triggers:
    2. Logon triggers:

1. What is CTE?
   * CTE is common table expression
   * It is a temporary named result set that can be referenced within a Select, insert, update or delete statement that immediately follows the CTE
   * Syntax:

WITH expression\_name(output columns)

As

(

Cte\_query

)

Select dep

from abc

join expression\_name

on abc.output=expression\_nmae.output

1. What is Savepoint in SQL
   * Savepoint is a command in SQL that is used with the rollback command
   * It is a command in Transaction Control Language that is used to mark the transaction in a table
   * Consider you are making a very long table, and you want to roll back only to a certain position in a table then this can be achieved using the savepoint
   * If you made a transaction in a table, you could mark the transaction as a certain name, and later on, if you want to roll back to that point, you can do it easily by using transaction’s name
   * Savepoint is helpful when we want to rollback only a small part of a table and not the whole table. Savepoint is like a bookmark in SQL
2. What is formcollection in MVC?
   * It will automatically receive the posted form values in the controller action method in the form of key/value pairs
3. File in .edmx file
   * Context file
   * Model file
   * Designer
4. IEnumerable vs IQuerable
   * IEnumerable is suitable for querying data from in-memory collections like List, Array and so on.
   * IQueryable is suitable for querying data from out-memory (like remote database, service) collections.
   * While querying data from the database, IEnumerable executes "select query" on the server-side, loads data in-memory on the client-side and then filters the data.
   * While querying data from a database, IQueryable executes a "select query" on server-side with all filters.
5. Skip and skipwhile
   * Skip specifies a number of items to skip.
   * myList.Skip(10)
   * SkipWhile allows you to supply a predicate function to determine how many to skip.
   * My.SkipWhile(x=>x.Age<18)
6. Var and let
   * var and let are both used for variable declaration in javascript but the difference between them is that **var is function scoped and let is block scoped**.  
     It can be said that a variable declared with var is defined throughout the program as compared to let.
7. Convert.toInt32 and int.tryParse
   * Passing a non convertible value to convert.toint32 will return 0 as output
   * Passing a non convertible value to int.tryParse will return a Boolean value of false
   * Passing a non convertible value to int.Parse will return a null exception
8. Count textbox in jquery
   * Var total=0;

$(‘input[type=”text”]’).each(function(){

Total=total+1

});

Alert(total);

* + Alert($(“input:text”).length)

Notes:

Boxing-> value type to ref

Unboxing-> ref type to value

All joins with proper examples

Boxing and unboxing  
Static Class and Singleton Class  
Garbage Collection, which namespace used.

Strongly typed view

Solid principles

Filter

Action method

How .Net framework works

.Net components

CTS

Magic table

Middleware in .Net Core