++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

DOT NET  
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

1. Assemblies and structures - fundamental unit of deployments and version control and reuse

Entry point - DLLMAIN, WINMAIN, MAIN

Security boundary, type boundary, reference scope, version, deployment,

1. CLR - runtime that manages and executes code in any language

Common type (value and reference) , Common language, Garbage collector, just in time compiler ( MSIL code to native code ) , meta data

Functions - converts into native code, handles exceptions, provides type safety, memory management, security, performance, platform independents

1. .net core vs .net framework

* Open source, cross platform, high performance, app development, collections of nuggets, micro services, no rest apis, no access security
* No open source, single package, only for windows os, no micro services, supports REST APIS
  + Managed code - get the services of managed runtime environments like CLR provides garbage collections, security …….
  + Unmanaged code - directly executed by OS depends upon computer architecture runs on platforms
  + ILDASM - IL disassembler

1. Console app - takes input and displays output at console with standard input, output, error

Class lib - code data used by other programs which can implemented in other

* Basics - optional parameter - should be at last of methods parameter in between gives CTL
  + Named parameter - name : “john”
  + Local functions - able to declare function inside the functions
  + Read only - can change value by constructor other wise cannot change the value
  + Destructor - only one to class not defined in structures cannot be overloaded or inherited destroy instance of class calls finalize methods
  + IDisposable : defined inside System class , need to notify GC it ignores that object during cleaning, if class has destructor only code in it of invocation of dispose, if base class has implement by it should not implement child class but child class can override dispose(bool) method
    - Methods = public dispose calls dispose(bool) method
      * Protected virtual Dispose(bool) all variable set to be null can override
      * Bool disposableValue - resource cleaning must not happen more than once
* Class - static classes only contains static members, not allowed to create object, sealed classes cannot be inherited, no instance constructor but static constructor
* Static local functions - after 8.0 if static function tries to access the scoped variable then compiler will give error
* Static class are sealed cannot inherit
* Access specifiers - public, protected, private, internal ( limited to current assembly), protected internal (current assembly and derived classes ), private protected (current class and child class within assembly)
* Constructor in inheritance - if does not have constructor of base class only member can be assigned but if has constructor then need to extend the constructor by : base(r, h)
* Overloading in inheritance -
  + cpp - no overloaded methods if find out method in derived class convert type and performs never distributes in scope
  + JAVA - works across scopes find correct version of methods
  + Csharp - gives preference to derived class first and do implicit casting to tackle this if we add function which will not be implicitly casted in derived class then compiler will check in base class for signature
* Method shadowing - (hiding) hide method of base class so that derived class will not override it using new object can access only new method
  + To call base class method
    - Base.method()
    - Casting while calling method
    - Using parent class reference to object
* Override - static non virtual method cannot be overridden
  + virtual - in defined base class Override - in derived class
  + without keywords use new in derived class but base class reference can access method
* sealed - sealed method cannot be overridden and sealed class cannot be inherited
* Abstract - method does not have body inside abstract class only
  + Abstract class can be created without abstract method cannot be static can have get and set methods
* Interfaces - cannot have private methods, default - public, abstract, loose coupling, total abstractions cannot explicitly use public and abstract keyword
  + Explicit interface implementation - when implementing method from two interface using name of interface. Method name is to differentiate
  + Inheritance
  + Default methods - method with declaration and definition to access method need to override can have private, protected…
* Operator overloading - conditional (&& || ), assignment (+=, -=, =) cannot be overloaded, comparison (==, !=) need to overload both
* IDisposbale - manages garbage collector destructor -> Dispose() stop running twice GC.SupressFinalize(this)
* IComparable - (comparable) takes one reference, compareTo
* Icomparer - two reference , compare
* IEquatable<Person> - equals

1. value type

* Struct - if used new then calls constructor if not same as class, on inheritance, no destructor, can implements interfaces, no abstract virtual protected
* Enum -
* Out - passing arguments to method as reference type method returns multiple values
* Refs - calling method by ref
* Nullable - to get value getvalueordefault() ?/ Nullable<data\_type>
* ?? null coalescing left ?? right - if left is null then right
* ??= - a ??= 200 if is a null then assign a = 200
* Array - jagged array - array of array each array can be different size
* Indices - ^1 in array size represent start from end, this[] keyword
* Range - … represents start..end

1. Generic class - type safe class <T>

* Constraints - where class?, struct, new(), notnull, base class, interface
* Generic collection - type safety, strongly typed, List<T>, Queue<T>, HashTable<t>, stack<t>
* Non generic - not strongly type arraylist, queue, stack, hashtable
* ICollection -isSync()
* IDict
* IList
* Tuple two parameter passed create method is created

1. Delegates - reference of method , encap, type safe, anonymous methods, needs to match signature, must be void
   1. Public delegate void name(int a) => name n = new name(10) => n(20)
   2. Multi cast invoke
   3. Func - get parameters and return a value, last parameters are considered out
   4. Action - get parameters and return nothing
   5. Predicate - check conditions one parameter and return true false

* Anonymous methods - inline delegate does not have goto, break, continue no out and ref
  + Like default methods in interface
  + Name p = delegate(string n){}
* Lambdas - (x) => {}

1. Checked error - enables overflow and conversion at compile time

Unchecked - overflow get ignored

Try, catch, finally

* User defined - Exception class : base (msg)
* Events - encapsulated delegate, two parameter and return void, source many subscriber
* Async calls - IAsyncResult async, reflection and then call the method
* Anonymous type - do not have name creation without defining them System.Object

Read only, no methods, only reference type, scope limited to methods, nested anonymous and linq using new {,,}

* Extension methods - add new method without compiling and modifying code need to be static
* Partial class - single class into multiple class same assembly and namespace, signature diff name
* Partial methods - one part in one partial class other in other class, void return type, private implicitly
* Linq to objcts - provide query language like to iterate over it, var -> iEnumrable (yield)
* Deferred execution - return latest data, execute when accessing
* Linq method - same as above but just function type
* Plinq - order record asParallel() ParallelEnumerable class

1. Reflection - obtaining info at runtime
   1. Custom attributes - meta data of extensions
      1. Using attributeusageAttributes -
         1. Attributetraget.all - all parts of program
         2. Inherited = true/ false -
         3. Allowmutiple = true/false -
      2. Defined attribute class : Attribute
      3. Defining constructor
   2. Invoke , getMethods , getType, Assembly.Load

* File I/O -
  + File.readALlText(), readAllLines(), stramReader(), writeAlltext, writeAllLines, streamWriter
  + System.IO.Directory.getCurrentDirectory, GetFiles, createDirectory, SetCurrentDirctory,
  + Stream writer - flush()
  + Stream reader - peek() and seek()

1. Threading -
   1. threadStart - new Thread(new ThreadStart(funcname)) , thr.start(), delegate
   2. thread pool - QueueUserWorkItem(fun)
   3. sync - consistency, no interference, lock(this)
   4. Monitor - provides sync objects more control over sync than lock Monitor.Enter(), Monitor.Exit(), Monitor.TryEnter()
   5. Async await - async -> async tasks(completes method without waiting for other method to complete) await->suspending async method unitl task completes

* Task parallel lib - higher abstraction System.threading.tasks
  + Tasks.delay, start(), wait(),
  + NotonFaulted, OnlyOnCancled, OnlyOnRanToCompletion, RunContinuationsAsync, LazyCancel, DenyChildAttach, PreferFairness - taskschedular

1. ASP.NET MVC

* Architecture - MVC page req, start, initlization, load, postback, rendering, unload
* Controllers - System.web.mvc.Controller : controller
* Actions - must be public, cannot contains ref and out, cannot be static, can be overloaded but not on signature but action name
  + Filter - Action method selector override method isvalidforRequest(controller context, methodInfo)
  + viewResult html , jsonResult to allow show jsonRequestBehaviour.allowGet, partialViewResult, ActionResult, RedirectResult, fileResult
  + contentresult return Content(“<h1></h1>”)
  + RedirectRoute, HttpStatusCode
  + HttpGet HttpPost - {fromBody} {fromcollrction), acceptVerbs , noaction will be just method and doesnot route to the method
* Views
  + Razor view - @ end with ;
  + Html helper - it is method can be reused in razor class @helper and in view @nameFunc
    - Standard - @html.textbox , @html.checkbox. @html.beginform endform
    - Strong - @html.hiddenfor, labelfor, checkboxfor,
    - Template - @Html.display
  + Viewbag same as viewdata but viewdata is dict and need typecasting
  + Data annotations - validation for model data, required, datatype, range, string length, display, maxleng. Bind, RegularExpressions
  + Client side validation @html.validationsummary(true)
  + Self validation - :IValidateableObject - validate set property to see the result at this place
  + Strongly typed view - @model IEnumerable<MvcApp2\_InternetTemplate.Models.comment> now you access comment in views
* State management -
  + TempData - data available until it is read when move from one controller to other controller it will be available to read but can we keep() to read for next method peek() access and keep it for next request
  + Session - like viewData access it @
  + Cookie - HttpCookie. - cookie.value - add -> checking if exists Cookies.allkeys.contains just like viewData
  + Query String - for get method, new {}
  + Application - stored in server same for all users and session no expiration date application lock unlock used same as viewData global.asax
* Module
  + Partial view - @partial view
  + Child action - can not be called from browser request this need to be called from view
* ADO.NET -
  + Connection <-> Data Source - beginTransaction() Server.creteObject, ConnectionString
  + Command - execute sql, procedure, DataSet, cmd.connection, cmd.CommandText, cmd.CommandType (text, table, procedure), ExecuteNonQuery (number), ExecuteReader (sqlDataReader) executeScaler (first column of first row)
  + Data Reader - SQLDataReader, executeReader(), Read(), GetInt32, getString(1)
  + DataAdapter - DataSet, fill() relations.add()
  + DataSet - (), (string name), (serializaionInfo, context)
    - Formatter - writing object in format
    - ObjectIdGenerator -
    - ObjectManager - while deserialized
  + Datatable - acceptChanges, Clear, Clone, Copy, newRow, RejectChanges, reset, select
* Routing - Route name, URL, Defaults, Constraints {controller / action / {id} <- routing table
  + RouteConfig.cs - refisterRoutes - IgnoreRoute, MapRoute{name, url, default }(mapcontollerRoutes)
  + Attribute - [Route()]
  + Life cycle - Middleware -> routing -> controller -> action -> view
* Bundle - scriptBundle, StyleBundle, DynanmicFolderBundle
  + bundleConfig.cs- RegisterBundlers(BundleCollection )
  + Bundle.add(new Scriptundle()) @scripts.render(“
  + Bundle.Add(new StyleBundle().include(“~/content/bootstrap.css))
  + Custom helper function
    - Inline inside view
    - External - html helper @using mvcCustomhelper.helpers
  + Handle error - [handleError][handleError = typeof(divebevbydzero) view=”error1”] -> handleErrorAttribute
  + Custom filter - ActionFilterAttribute (onActionExceuting(actionexcutingcontext)) -> filterConfig.cs - RegisterGlobalFilters(GlobalFilterCollection ) --- filters.add(new Filter)) -> [Filter] on controller or on methods
* MVC Security -
  + [AllowAnonymous] web.config -> <authentication mode=”forms”> <forms loginUrl=”/” ViewBag.Title = “Secured” -Secured.cshtml
  + [ValidateAntiForgeryToken()] - authority
  + Cross site Scripting - uses web app to send malicious code [validateInput[]
* Entity Framework
  + Code first - DbContext(DbContextOptions options) -> DbSet<Employee>
    - Connection string in Startup.cs ConfigureServices(IserviceCollecion) -> migration CreateTable - Up() and DropTable - Down()
  + Annotations - Data annotations - Key, Required, minLength, MaxLength,
    - Attributes - Table, Column, Index, ForeignKey , PrimaryKey
  + Fluent api - DbModelBuilder - override onModelCreating
    - ComplexType(), HashIndex(), HasKey(), HasMany(), Map(), ToTable(), IsRequired, HasColumnName()
  + Migrations - Database schema in sync with entity timestamp\_name.cs (up and down methods ), timestamp\_name.designer.cs (metadata) , contextClassModelSnapshot.cs
    - DropcreateDBAlways
    - DropCreateDBWhenModelchanges
    - CreateDatabaseIFNotExists
  + Crud - CrudContext
    - Getall - Employeess.list()
    - Add - Employees.add() savechanges()
    - Edit - SingleorDefault(e -> eid) 🡪 employee.state = entitystate.Modifed
    - Get - SingleorDefault(e -> eid)
    - Delete - SingleorDefault(e -> eid) 🡪 remove

1. Localization - customization to make our applications behave depending on culture and locale 🡪 Resource.en-GB.resx , Resource.mr-IN.resx
   1. Thread.CurrentThread.CurrentUICulture.Name == “en-GB”;
   2. New CultureInfo(“en-GB”)
2. Web API

* API - ApiController
* CORS -
  + Services.addCors(options => options.addDefaultPolicy(builder => builder.withOrigins
  + App.Usecors() app.useCors(builder => builder.AllowAnyOrigin()
  + ConfigureServices(IServiceCollections) 🡪 services.addconfigure
  + Configure(IApplicationBuilder app, IWebHostEnvironmet env) 🡪 app.UseCors(builder =>
* Consuming using a Client
  + Client = new HttpClient()
  + Client.Bas
  + eAddress
  + Client.DefaultRequestHeaders.Clear() 🡪 Accept.Add 🡪 HttpResponseMessage client.GetAsync 🡪 Res.content.ReadAsStringAsync().Result()
  + Address, binding, contract ->service, data, message
  + Types of binding system.servicemodel
    - Basichttp - http asmx
    - Netmsmq - message queue
    - Nettcp - tcp
    - Netpeertcp - peer to peer
    - WSHttp - https secure
    - WSDualhttp - bi dire
    - newNamedPipe - most secure used pipes
* NewtonSoft APIs
  + High-performance JSON framework
  + Serialize JSON
  + Deserialization - JsonConvert.SerializeObject(product) 🡪 JsonConver.Desera
  + JsonSerializer - New JsonSerializer().Converters.Add()
  + Linq - JArray 🡪 JObject 🡪 o[“MY”] = o.ToString()