**Why**

* Cross Platform

: Developed and Deployed on Mac , Linux Or Windows

: 70 % of deployment happens on Linux. No Licensing Cost .NEt doesn’t support Linux and Mac.

* Cloud Provider :

: Scalable : On demand Resource

1 Server : 8GB … : Application => 10000 User

Sat : Sun : 1 Lakh : 1 Server 32 GB : 1 Lakh

* JS Frameworks : Angular React,Vue

: Performance : Swiggy , Zomato

* Docker and K8s

: Application is hosted in K8

* Reduced the infrastructure cost : 4 .. 2 Server

**.NET :**

* 20 Years : 1998
* It was not created keeping Cross Platform, Cloud , JS frameworks, K8 : Popular

**.NET Core**

* Cross Platform : Linux, Windows and MAC
* Cloud ready :
* JS Frameoworks : Angular React
* K8 and Docker : Reduced the infra cost

1 Server: 24 GB RAM

VM 1: 5GB : App

VM2 : 5GB : API

VM2 : 5GB : DB..

HostMachine: 9 GBRAM

Docker : APP + API + DB : Memory Share

**What**

* .Net Core is framework for building an cross platform application

**When**

* .Net Framework Or .Net Core

1. Cross Platform : .Net Core
2. Performance : .Net Core
3. Docker : K8 : .Net Core

**.Net Framework**

* Application : Third party libraries , Packages Only .Net Framework
* CAS, Remoting

.**Net Core Version**

* 1.0 ,1.1
* 2.0 , 2.2
* 3.1 : Stable : Visual Studio 2019
* [Download Visual Studio 2019 for Windows & Mac (microsoft.com)](https://visualstudio.microsoft.com/downloads/)
* 5.0 :

**IDE :**

* VS 2019 : Mac Windows
* Visual Studio Code : Cross Platform , Lightweight

.**Net Core 3.1**

* ASP.Net Core Web App
* Console App .Net Core
* Windows Form : Was not present 2.2
* WPF : Was not present 2.2

.**Net core 3.1 Languages Supported**

* C#, F# and C++

.**Net Core 5**

* VB.Net Support
* EF6 Support Removed.

ASP.Net and Web Service => ASP.Net Core or Web API.

ASP.Net Core or Web API => Rearchitected

* Application structure new

Testing : XUnit, Nunit Project … : Nsubstitute , Moq ;Packages

**Demo : How to build project in .Net Core**

* Project Structure

1. Program.cs : Entry point for your application. Application Set up
2. Startup.cs : Execute the Request using Request Pipeline. Activate the Objects

Required.

1. Configure

* Build or Reprsents Request Pipeline
* Contains Middleware

: Code that get executes in the pipeline

* IApplicationBuilder : Application Object is Created using this
* IWebHostEnvironment : To identify the environment in which code is running .

1. ConigureService

* Builds Service Object : DB , Security
* Object is Provided to Request Pipeling
* IServiceCollection

: Service Object Activate

1. appSetting.json : Configuration of your application: connectionstring … Keys
2. launchSettings.json : Application Startup Settings. Port number, Environment
3. Dependencies : Project Requirement, Packages , framework etc

Startup.cs

Program.cs

IServiceColl

IApp

ConfigureService

Configure

Req

Browser

Env

Home

DB

Middleware

3Middleware

2.Middleware: ne

1.Middleware : next

Request Pipeline

ASP.Net Core Platform

**To Create a middleware**

1. Use : Code , next : next middleware to be called
2. Run : Code , it wont contain next parameter

**Async or Await**

Server

Thread 1

Req : DB : 5

Browser

Request

Thread 2

Thread

Req : File : 10

Req : Web : 15

Thread 3

Middlewares

* software that's assembled into an app pipeline to handle requests and responses

Types

1. Built In Middlewares

* StaticFiles
* Response Caching
* Authentication
* Authorization
* Routing
* Endpoints

**Demo : Static File Middleware**

* Use to Serve Static Content : HTML, CSS, JS …
* Needs to be part of special www root folder
* These files are available without any authentication
* No authorization checks are performed on the static files.
* Static files served by the Static File Middleware, such as those those under wwwroot, are publicly accessible.

1. Custom = Later .. After MVC Framework

ASP.NET MVC Framework.

* Architectural Pattern

1. Divides application into 3 Parts
2. Model

Application

1. View
2. Controller

Model : Classes that holds data .

http://localhost:5000/home/index

Controller : Handles Req From browser

View : Display : UI

Adv

1. Separate different aspect of the app :
2. Loosely coupled
3. Helps Manage the Complexity
4. Unit Testable : Improves the Quality of code, it also makes sure your code is bug free.
5. Front End Designer they can separately work on the view Section

ASP.NET MVC: NO Configuration .. Convention

* Naming Conventions

Example : ASP.NET : HomePage : Content

ASP.NET MVC = HomeController => Name should always ends with the controller

Customer.asp : All

* CustomerController => Add Custom,Delet

Controller

* That handles request from the browser
* Talks to model to get data if required
* Pass data to View
* Pass that view to the browser /user
* Controller should inherit from Controller class

Rule

1. Controller Name Should end with “Controller” Word
2. All the controllers needs to be present in Controllers folder

Controller Contains Action Methods

* Action Methods Return Type is IActionResult
* IActionResult : It can Return multiple types

URL : [http://localhost/home{controller}/index{actionmethod}](http://localhost/home%7bcontroller%7d/index%7bactionmethod%7d)

<http://localhost/home/aboutus>

<http://localhost/home/contact>

<http://localhost/customer/index>

Setting Default URL :

// Default URL : http://localhost/home/index

endpoints.MapDefaultControllerRoute(); // routing ..

View

* Contains your display logic
* View Should be present insides Views folder
* Its compartmentalized by controllers foldername.

How Controller Can pass Data to View

Customer Index

RaZor Syntax

@ViewBag.CustomerId

1. ViewBag : dynamic

Delhi

Rohit

1

ViewBag

ViewBag

* Intellisense
* // For Passing object ViewBag is not Good Option
* // Best Practice : Create a model class

Model Approach

* We build a Class
* Class is made available to View
* To Access customer model in this case in the View . we need to Import that model into our view

@model modelName

* Create a Special Object Model
* We cant not pass 2 models in a single View. But if in case my view wants to display information from multiple models then we have to use a concept of ViewModels

Razor Syntax : is use to display information on View.

* It starts with @ symbol
* You can use all C Sharp Syntax : @if..else , @foreach, @for, While

[Razor syntax reference for ASP.NET Core | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-5.0)

Case Study : Application : Admin

* Product Information : Store in Memory Collection

// Read : Display

// Dummy List:Models

Best Practice : Repository Pattern

1. Contract : Interface : What all Methods will be available : Methods Add : application more loosely coupled
2. Implement those interface : Code

IStoreRepostory

Home Controller

IStoreRepositrou s =

s.Products()

ProductInMemoryRepo

ProductSQLRepos

ProductOracleRepos

Dependency Injection

* We use Mechanism of Constructor Injection
* It makes your application loosely coupled
* It makes your code unit testable

1. Interface : Istorerepo
2. Class who implement those interface: ProductInMemroy
3. Controller Use Ctor Injection Add to pass Dependent Object

private readonly IStoreRepository \_repository;

// Ctor Injection

public HomeController(IStoreRepository repository)

{

\_repository = repository;

}

1. Associate interface with the client he is pointing to.

services.AddScoped<IStoreRepository, ProductIOracleepository>();

* CRUD :Operation
* Pagination
* Connect with SQL Database