**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

Model

Startup.cs

Program.cs

?cultureinfo= de-DE = Germany

IServiceColl

IApp

ConfigureService

Configure

Browser

Env

Home

DB

2.Custom Middlewar

Endpoint

Router

1.Middleware : next

Razor

@

TagHelper

View

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

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

**ViewBag : dynamic**

Delhi

Rohit

1

ViewBag

**ViewBag**

* No 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
* DI concept was present before ASP.NET Core as well. Third party libraries Deve use to use for the DI

Unity,Autofac

* .Net Core Microsoft introduced his own DI injection mechanism

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

**Demo : DI**

* Explain you Service : Service LifeTime

Service : Generates a Random Number

1. Interface :Method
2. Class : Interface
3. Ctor Injection
4. Conifgure Service Define

Application

IRandomService

HomeController

: Index

RandomService

RandomWrapper

IRandomWrapper

1. Add Scoped

RandomService : 1000

HomeController

Request : Rohit

RandomService : 2000

Object

RandomWrapper

RandomService : 10000

Request : Swapnil

1. Add Singleton : For all Request Single Object

Request

RandomService : 12000

Request

1. Transient

* Everytime DI resolved , new object is created
* If we need any service for short span of time

**Memory Consumption**

* No Option
* Scalability : Cloud, K8s . .

Load Testing

* Infra .. Monitoring Tools
* Modernize : Cloud.. Autoscale ..

**Home and Customer Controller**

Problem

* I want to have consistent look and feel to all my pages.

Solution

* MasterPage : Consitent look feel

How?

1. Add Layout Page = MasterPage
2. This pages needs to be created in special folder. “Shared”

@RenderBody() : Where Child Pages Are loaded = ContentPlaceHolder

* ViewStart : This is Special Where we can configure MasterPages for all the views.
* We can also add Master page information specific to View
* You need to go to specific view and add ,

@{

Layout = “AdminLayout”

Problem :

* Every View We need to define entire path of the class, in order refer it

sample\_app.Models.

Solution :

\_ViewImports : All classes that you need to use inside view, the path should be mentioned here.

Problem : I want to see information of Particular Product

Solution

1. Page : Details
2. Link : It should navigate me to that same page

**TagHelper**

* Enables Server side code to participate rendering and creating HTML elements in Razor files.

Anchor Tags : <a asp-action="" asp-controller=""> @p.Name</a>

Form

* Textbox

Controllers

1. Add Package : Microsoft.ASpNEt.MVC.TagHelpers
2. Configure

* ViewImport file
* @addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers(AssemblyName)
* \* : From Microsoft.ASpNet.Core.TagHelpers .. all the taghelpers should be available to every view

**TagHelpers**

1. Built in TagHelpers

* [Tag Helpers in ASP.NET Core | Microsoft Docs](https://docs.microsoft.com/en-US/aspnet/core/mvc/views/tag-helpers/intro?view=aspnetcore-5.0)

1. Custom TagHelpers – Will do it Later

* Something which is not available in Built in needs to be created by developer

Adv

* **An HTML-friendly development experience**
* rich IntelliSense environment for creating HTML

Smart Urls : http://localhost:5000/home/details/1

View Data => Controller

Problem

* I want to Add a new Product

Steps

* View : Form
* Action Method : Model class Pass

Problem : zh-CN => China .. de-DE => Germany

http://localhost:5000/?cultureinfo=zh-CN

Browser

READ =>cultureinfo

OP : Germany

OP : China

English US / Germany

Solution

* Class : CultureInfo : He knows how to convert zh-CN to China

Global Code => Read Cultureinfo => Country information

**Custom Middleware**

1. Class : RequestCultureMiddleware
2. Take ctor Parameter : RequestDelegate Object => it calls next middleware
3. It has InvokeAsync Method : HttpContext : Current Request + Response .. class as parameter

How to call this middleware

* Whenever we want to add a method into existing class or interface. We use Extension methods

Problem : I am able to add product with null or invalid Values

Solution : Validation

Model Validation

* It makes sure that only valid data will be added to the database.

ASP.NET Core

Data Annotations : Attributes

1. Required : Validates the field is not null
2. Range : Specified Range : 18 to 50
3. Compare : Ex Password and Confiirm Password
4. EmailAddress : …
5. Phone : ..
6. CreditCard : ..
7. StringLength : Validates the Field for specified length limit
8. RegularExpression :..
9. Remote : Validates input on the client Side by calling action method on the server

For Example

Registration form , we pass a username and some validation happens , it says username is already taken

When We add Annotations we received an Error in the form of ModelState. IsValid Or Invalid .

Its app Responsible to check whether its valid or not

Best Practice

DAta

ViewModel

View : Create

Name

ProductEditViewModel

Validation Logic …

Product Model

Create View ID NOT Required for Update View ID Required

Names

Price 0,999

Description : 20//100

Data

View : Update

Product Name

ProductViewModel

Validation Logic …

View : EditByAdmin

Product Information

Remote Validation

* Implements Client Side Validation
* Calls Server Side Method
* This method has to be part of the controller where you want to add validation

Server Side

Method

Remote Validator

Action: method name

Controller = which ctrl

JSON : true/false

Demo : Check Category Textbox user has entered Chess/Cricket/Soccer

Swapnil … this username is already der

ViewModel

1. Validation Logic : ViewModel
2. Present Data in view different

Conversion : AutoMapper

ViewModel

Name

Cost

Name :View

Cost

Model : Model

ID

Name

Cost

Category

VM

Name

Category

Name

Category

Problem

* We need to Convert one object to another

Solution

* Use AutoMapper

AutoMapper

* A convention-based object-object mapper

1. Automapper
2. Automapper.DependencyInjection
3. Activate Services Related to Automapper
4. Create a class That contains logic for Automapper , this class needs to be inherit from the Profile class
5. Automapper logic should be present inside ctor . you need to use CreateMap Method
6. Use IMapper Interface , it contains Map Method that will call Automapper logic

Adv

* Improves Developer productivity
* Maps One Object to another Object automatically.

Problem : Application is deployed or hosted (Production) and suddenly client starts complaining its breaking.

Solution : Logs

Logging

* ASP.Net Core doesn’t include logging provider for writing logs to files.To Write Logs into files we need to use Third Party Logging Provider.

Logging Infrastructure

Third Party

NLog

Serilog

File.txt

Application

**NLog**

* Open Source and Flexible framework for logging
* NLog can write a log to one or more targets at the same time
* It can write upto 30 Targets at a time.

[NLog properties with Microsoft Extension Logging · NLog/NLog.Extensions.Logging Wiki (github.com)](https://github.com/NLog/NLog.Extensions.Logging/wiki/NLog-properties-with-Microsoft-Extension-Logging)

1. Install Libraries

NLog : Core

NLog.Web.AspNetCore

1. Build Configuration file : XML : Contains Settings of how to log

* Format
* Where to Store a file
* What all to be logged

Nlog.config

1. Configure Application to use NLog : File .. Whatever ILogger is wrting same thing , he is writing in Files

webBuilder.UseNLog().UseStartup<Startup>();

1. Write a Logging Code using Interface ILogger<T> : Microsoft : He doenst know how to log into Files

* Contains Methods : LogInformation, LogError, etc : By default Write into Console

Problem : Whenever application throws application , I am getting an Exception Page that contains StackTrace or everything related to that exception. This By Default Page is also called as DeveloperExceptionPage.

Solution : Custom Error Page So that user wont get DeveloperExceptionPage

Demo : Custom Error Page

1. Create Controller
2. Design Error Page
3. Configure Pipeline to use Custom Error Page in Staging or Production environment

Problem : When user may mistyped urls , we are getting default error page from browser

Solution : Can we get a default error page from ASP.Net Core

* Additional method of handling an error that can handle error codes like 400 to 599
* Status Code Pages : Middleware that handles errors between 400 to 599.
* This is Designed to handle cases where some part of the request went wrong without throwing an exception.

Different Error Handling Mechanism

[Handle errors in ASP.NET Core | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/error-handling?view=aspnetcore-5.0#usestatuscodepages)

1. ModelState
2. Exception Page
3. StatusCode
4. ExceptionFilter.

Routing

* Responsible for matching incoming requests and dispatching it to app executable endpoints.
* Endpoints are Request handling code
* Endpoints can extract values from URI

Routing Middlewares

1. Use Routing : Middleware Pipeline , it looks for Set of endpoints defined in the app and select best match on the request.
2. Use Endpoints : Contains Endpoints, it runs delegate associated with the endpoint

Type Of Routing

1. Conventional Routing: when you write Routing Logic in Startup.cs that is called as Conventional Routing

Code

EP1

UseEndpoint

UseRouting

https://localhost/Chess

EP2

EP2