**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

Entity Framework

?cultureinfo= de-DE = Germany

IServiceColl

IApp

ConfigureService

Configure

Browser

Env

Filters

logic

Home

-Index

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

https://localhost/Chess

UseRouting

EP2

EP2

1. **Attribute Routing**

* You can Route attribute to specify Route for every action.
* ASP.NET MVC5 they introduced routing.
* It allows you to define routes in the same file as controller.
* It help us to make our Start up file to look bit less messy.

**Routing Constraints**

* Add some Validation on Routes

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-5.0#route-constraint-reference>

**Custom TagHelper**

Problem

* We are not able to see all products.

Solution

* Add Pagination
* Build dynamic buttons :HTML
* HTML Control => Pagination 1.2.3.4..5..
* Build my own HTML control that can generate number of buttons based on the products count.

HTML for Loop .. 1..2..3 = He should know how to work with Routing infrastructure

* Integrate with Routing

Build Custom TagHelper : Pagination

1. We need someone who can store information about TotalItem, ItemsPerPage, CurrentPage, TotalPage.
2. Create a class PageLinkTagHelper it should inherit from the TagHelper class

Problem

* Display Categoriwise information but user should not type URL.

Solution

* Create Links that will load information Categorywise.

View Components

* Similar to Partial Views
* Works with Razor Pages
* Randers chunk rather than whole response.
* Typically invoked from Layout Page.

Problem

* Number Of links generated doesn’t match with number of products available per category.

Solution

How

1. Create a class NavigationMenuViewComponent : ViewComponent
2. This class should hold Invoke Method

Completed

* Pagination
* Reading
* Creating
* Deleting
* Updating

Problem : How to Connect our application to Database.

Solution :

* Microsoft SQL Server.

Entity Framework For the Database Connectivity

History

* ADO.NET : Use ADO.Net For Connecting application with Database

SqlConnection : ConnectionEstablished

SqlCommand : “Select ProductIds,Names, Cost From ProductInfo”

Cmd.parameters.Add(“ProductIds”,@ProductId)

Cmd.parameters.Add(“Name”,@Name) .. 20 lines of Code

Cmd.ExcuteReader,ExecuteNonQuery

Iterate .. show the data

Problem

* Run time errors. : No run time Error
* Intellisense : You will get intellisense
* Lot of Code we need to write to map : 3 line.

Adv

* Time Save : Developer Productivity

What is Entity Framework

* “Entity Framework is an object-relational mapper (O/RM) that enables .NET developers to work with a database using .NET objects. It eliminates the need for **most of the data-access code** that **developers usually need to write**

Adv

1. Need to Write Less amount of Code
2. Intellisense
3. Less Runtime Errors.

Object Relational Mapping Framework : Entity framework

Relational

Objects

1.TataPowerDBDataContext

* ConnectionString

Application

Sql Server

DB : TataPowerDB : Products

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Name | Cost | Category |
|  |  |  |  |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Change Tracking

1. Product

* Id
* Name
* Cost
* Category

Tables = Classes

Table Fields = Classes Properties

DataContext = Database

1. Install Required Libraries

* Microsoft.EFCore
* Microsoft.EfCoreSqlServer

1. Data Context

* Tracks and makes the necessary changes in the database
* ConnectionString as Parameter.
* Methods required to perform changes in the database

1. Define ConnectionString in appSettings.json Done
2. Pass this ConnectionString to DataContext : Startup.cs Done
3. Code First Approach : Classes => DB and Table Generate Done

* Need to use dotnet commands
* Create Migrations

dotnet ef migrations add initialmigration(migrationname)

File Providers in ASP.NET Core

* Use to Give access of File System to your application.
* Internally some components they use File Providers

1. Static Files
2. WebHostingEnvironment
3. Razor Pages

* It wraps System.IO , it provides access to physical file system

Folders

Files

File Providers

Application

Physical File Providers : Access to your File System

Need

Application

Application

* When 2 applications wants to communicate with each other on internet.

Microsoft

1. Remoting

* Calling Remote function or Remote procedure call.

Format

* Binary Format

Adv

1. Performance

Problem

* Both has to be built on the same platform . Both should be .Net.

1. Web Service

* Web Service Description Language

Format

* SOAP format communication(XML)

Adv

* Two application built on different platforms can communicate.

Problem

* Slow. Heavyweight format it uses.
* WS- Security,WS-Attachment … WS\*

<Name> Sachin </Name>

1. Web Service Enhancement

* Web Service Description Language

Format

* SOAP format communication(XML)

Adv

* Two application built on different platforms can communicate.
* WS\* Supported

Problem

* Slow. Heavyweight format it uses.

<Name> Sachin </Name>

1. WCF : Windows Communication Foundation

* Microsoft Technology for the implementation of Service Oriented Architecture.

Formats

Binary , SOAP

Adv

* Use to support multiple formats. Binary , SOAP
* Performance

Problem

* Lot Of Configuration

Server

1.Home

Browser

Ang/React

Ang/React :JS

Client Side :Ang

1. Performance
2. Usability

Problem

* Initial Load will take time

Problem

1. For every request you need to go to the server
2. Performance
3. Usability
4. Web API

* Use to build HTTP Services.
* Accepts HTTP Requests and Generate Response that contains data.
* Provide access to an Application Data.
* Use to Provide rich client applications(Browser) with data.
* Server Side application can use Web APIs.
* Default format for Web API is JSON : Javascript Object Notation
* Combination of URL and HTTP Method Describes an operation to be handled by Action Method

Application

* Angular
* React
* Blazor
* Ember
* Mobile Application

Web API : Controller : ControllerBase

HTTP METHOD Or verb: GET,POST, PUT , DELETE

Action Method : GetProducts()

HTTP METHOD : GET,POST, PUT , DELETE

Action Method :GetPRoductById(int id)

JSON :Data

-

HTTP Verbs

1. HTTP GET : Retrive Data object
2. HTTP POST : Create a new Object
3. HTTP PUT : Update an Existing Object
4. HTTP DELETE : Delete an Object

ControllerBase

* API Controller is Derived from Controller Base class. It provides access to features provided by MVC framework.
* It gives access

1. HttpContext
2. ModelState
3. Request
4. Response
5. RouteData
6. User

Controller Attribute

1. Route : URL of the Controller is Specified using Route Attribute
2. HTTPGET
3. HTTPPOST
4. HTTPDELETE
5. HTTPPUT
6. APIController : Advanced Feature inject into Controller class. This feature are related to ModelState.

Rule Of Thumb

1. GET : Read
2. POST: Insert
3. DELETE : delete
4. PUT : Update

How to Test APIS

1. Browser : You can only test methods with GET attribute
2. POSTMAN

* Third Party Software , Free . Use to Test API
* We can test API thoroughly.
* Local Testing : During Development.

1. Swashbuckle

* Library you can include in your application that create a Open API that can be use for the testing.
* WSDL

I .Install Swashbuckle

1. Enable Swashbuckle in Startup class

// Enable Swashbuckle

services.AddSwaggerGen(options =>

{

options.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo

{

Title="Tata Power Open Web APIS" ,Version="v1"

});

});

1. Include Swashbuckle in Request Pipelin

app.UseSwagger();

app.UseSwaggerUI(options =>

{

options.SwaggerEndpoint("/swagger/v1/swagger.json", "Web App");

});

1. gRPC

* 7… 10 Times Faster than Web API.
* Supported by many platforms.. C# java , Python GO
* Developed by google in 2015

Binding Source

A binding source attribute defines the location at which an action parameter's value is found.

FromRoute - in the route url

FromBody : request body

FromHeader : Header :

FromService : ILogger Service.

Web API => Database

1. Data Context = > DB and Perform Db Operations
2. Install Libraries
3. AppSetting : connectionString
4. Passed the connectionstring to DAtacontext
5. Repository Pattern : Interface + class

Best Practice 1 : Try to Make Web API Methods Asynchronous : DB Call , Network File Reading

Why??

* ASP.Net Core Platform processed each request by assigning thread from the pool.
* Number of Request processes is totally depend on size of the pool. It might possible that ASP.NEt Core Request thread might run out of threads they might have to process request.
* This problem can be addressed by defining async actions. It allows thread to process other request when there is blocking request they are handling.

Best Practice : Prevent Over Binding

Problem : For no Reason , ProductID = 0 is passed during every SaveProduct Request. : Over Binding.

Solution : Prevent Over Binding : Need to Pass only the things that is Required.

Class : Name, 5 Fields Pass

Convert .

1. Automapper : .. Done
2. Method : That can convert your Object into some other Object..

After MVC .NEt Core 3.1 . Model State Validation Errors are automatically handled.

Front End === > Web API

1. Save Product => + ve Response , Data is Added

* If I pass wrong data then Data is not proper

1. GetProductById = -1 > ID not found

* This is your data

Action Results

* MVC Framework Sets the Status code automatically.
* Action Methods can direct MVC framework to send specific response by returning object that implements IActionResult interface. This allows Action method to specify the type of Response Required.

Following are ActionResult Methods

1. Ok => 200 + data Object (Optionally)
2. NoContent : Produces 204 No Content Status Code
3. BadRequest : 400 Bad Request Status Code. Accepts model state object that describes problems to the client.
4. UnAuthorized() :
5. NotFound : 404 Resource not found.

Best Practice 3: Always send Some sort of Status Codes along with Output. It is very useful in the client side development.

Projecting Selected Propeties

* Returns only the properties that client requires.
* It gives control over each response.
* We use Anonymous object for this.

Best Practice 4: Send only which is required by client.

Problem : By default Web API always returns JSON . along with that at specific instances I may want to pass data in different format. For example XML.

Solution : Content Formatting and Content Negotiation.

**Content Formatting**

* Web API supports multiple formats that action methods can produce.
* The content format selected it depends on Different factors

1. Formats the client can accept.
2. Formats the application can produce.
3. Content Policy specified by action methods.
4. Type Returned by the action Method

Web API

* JSON

XML

[XML]

GetProduct[]

Client

JSON

**Content Negotiation**

* Client include Accept Header in request that specified format that client willing to receive.
* Client format in Accept header not at all impacting the output

**Fully Respecting Accept Headers**

* Problem: Whenever we ask for a format from client and if Web API doesn’t support that then , it will send by default JSON format. If you want to override this behaviour you can use Full Respecting Accept Header concepts

Solution

* We need to configure 406 Error , It means Format Not Available.
* It overrides the behaviour of Web API of Sending By default JSON format.

**Best Practice : Try to Add 406 Error if The format is not supported**

Specifying Action Result Format

* Data formats that MVC framework can use for an action method result can be constrained using Above method.
* Produces Attribute should be used for that.
* Consumes Attribute to Consume Data in Specific Format.

**Best Practice : Define Format Action Result on Action level. It improves Readability of code .**

Problem

1. SaveProduct : Ok Or BadRequest.

* Third Party Application wants to consume your Web API
* We want to give the information about all possible return types of your APIS

Solution: ResponseType : Best Practice

[Client IP safelist for ASP.NET Core | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/security/ip-safelist?view=aspnetcore-5.0)

Http

Library

Client Side

Angular/React/jQuery

http://localhost:5000

Web API

Server Side

C#, Java.. HttpClient

Demo : jQuery ==🡺 Web API

* jQuery => Ajax : Web API
* Type : GET/POST/DELETE /PUT
* URL : Url to which request needs to be send

$.ajax(type:”GET” , url :” <http://localhost:5000>”, success : function(data ){ display data}, error : function(err) {display error})

jQuery Official Site Link : [jQuery](https://jquery.com/)

* jQuery always runs when DOM is ready.

CheckWebAPI:1 Access to XMLHttpRequest at 'http://localhost:5001/api/Products' from origin 'http://localhost:5000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

**Cross Origin Resource Sharing**

* Browser Security Prevents a web page from making requests to a different domain than the one that served the page.
* CORS is not security feature, w3c standard.
* How to Enable such cors in ASP.Net Core

Web API2 => Web API Core

ASP.Net Core Identity

* Extension to ASP.Net Membership that we use to use.
* API that supports UI login functionality.
* Manage users, passwords, Profile data , roles , tokens.
* SQL Server can be use to store identity information.

Secure an Application

1. SQL Server
2. Azure AD / Windows AD

Tata : SignIn => Single Sign On : Attendance Application, Leave Application , Salary application.

Microsoft.AspNetCore.Identity

1. UserManager
2. SignInManager
3. Identity User

SQL

App

AD

User Manager : Manage the Users

SignInManager : Login Logout

IdentityUser : Represents a single User

IdentityRole : Represents a Role

1. Install ASP.Net Identity Package. : Microsoft.AspNetCore.Identity.EfCore
2. Need to add class that represents a User
3. Need to modify context class to user IdentityContext

* Configure our application to use Identity Management System.

1. Generate tables required to Store Identity :
2. Generate Roles

Problem : Authorization : Want to Create Couple of Roles.

1. Administrator
2. User

Demo : How to do Registration of User

Registration Form

FirstName

LastName

DrivingLicense

Email

Password

ConfirmPassword

1. ViewModel

* RegisterViewModel

Model

ApplicationUser

Login

* AccountController We need one action that Returns Login View

1. Login Returns View --

* LoginviewModel that view accepts.

1. Login(Model ) And performs the Login --

* SignInManager should be use to Perform Login

ASP.Net Core Application is Authenticated and Authorized

Web API

* JWT Authentication

Json Web Token

1. Request : Login

Web API

* Authentication
* Authorization (JWT)

Login : Authenticate : Return JWT Token.

Index

Create

SPA/ ASP.Net Core

2. Req : Index + JWT Token

Token

* Header : Alg
* PayLoad : Content :

: Identity Of User

* Emails,UserName .. etc : **Claims**
* Signature
* Encypt

Browser : JWT Token : ExpiryTime

Header :- Contains the information about type of Token and the name of algo.

Payload : Contains User Related information , like userId, Email, Role.

Signature :- Server uses this signature to verify whether token contains valid information, Its digital signature that gets generated by combining the header and payload together . its totally based on the Secret key which only server knows.

Demo : How to Perform JWT Authentication Mechanism.

We need to first enable Identity for our Web API

1. Libraries

* Microsoft.AspNetCore.Authentication.JwtBearer --
* Microsoft.AspNetCore.Identity.EFCore --

1. Login

* Create class ApplicationUser : IdentityUser --
* DBContext => Replaced => IDentityContext --
* Need to Read ConnectionString and Build Configuration Object --
* Enable Identity For Web API --
* AccountController , Login API => User is Valid

If the User is Valid

1. JWT Token Generation Code.
2. Configure your application to use JWT token or Enable all the objects required for JWT token.
3. Issuer : Actual Server who created the Token.
4. Audience : Receiver of the token.
5. LifeTime : When the token will be expired.
6. Key : Valid Key
7. Generate the token

Filters

* Request is processing we can inject our logic.
* Allows us to run certain actions before or after specific stages of the Request Pipeline.

OP

Filters

Logic

Before Action Method Or After Action Method

Request

How its different From Middlewares

* Excecution of middleware occurs before MVC context .
* Middleware doesn’t have access to Actions and its context.
* As model binding has not occurred yet. Validation of values cant be possible through middlewares.
* Middleware run on every request regardless of which controller or action called.

Need

1. Logging
2. Avoid Duplication of code.
3. Caching
4. Error Handling

Types of Filters Available

1. Authorization Filters : [Authorize (Roles = "Administrator")]

* Run First and Determines the Authorization.

1. Resource Filters : Built In Filter [Attribute]

* Use To Implement Caching.

1. Action Filters :

* Filter Executes before and After Action Execution
* Can be applied only on controllers or actions.

1. Exception Filters

* Use to globally handle all unhandled exceptions of individual requests.

1. Result Filter

* Use to Alter the action result.
* Any last minute changes you need to do . you can use Result Filter.

Example : Add Some Headers to the results

Demo : Action Filters : Avoid Duplicated Code, Execute Some logic before or After the Code gets executed : Logging

* Create Class
* Class Inherited from ActionFilter Attribute
* It has 2 methods

1. OnActionExecuting :- logic : When the action execution is in progress
2. OnActionExecuted : logic: When the Action execution gets completed.

Exception Filters

* Use to handle the unhandled exceptions that occur in an application.
* They don’t have before or after methods
* It Implements OnException Method
* Method will be automatically called when unhandled exception occurs.

Demo : Exception Filters : If Product Price entered is equal to 0 , it will trigger Exception Filter.

1. Create class
2. Class : IException interface
3. Implement OnException Method

* Contain the logic to navigate exeption to custom error page

1. Register the filter
2. Apply it.

Areas

Naukri.Com

* M : Job Seeker = Functions => Search, View, Request Raise. .. 30.. 40 functions .. 10 controller .. Views
* M: Employer = Functions => Post,Register,Search based on criteria .. 30.. 40 functioons .. 10 controller .. views
* M: Administrator = Functions => Manage Job Seeker, Employer .. 30.. 40 function .. 10 controller … views .. 120 Views

Problem : Developer to manage this code base.

Solution : Modulewise

* Controller
* Views
* Models
* Layout

Areas

* ASP.Net Feature use to organize related functionality into group as a separate.
* Provide a way to Partition ASP.NEt Core Web app into Smaller functional group.

Functional Groups

1. Razor Pages
2. Controllers
3. Views
4. Models
5. Layout

Example

When to go for Areas

* App made of multiple high level functional components that can be logically separated.
* Partition app so that each functional area can work independently.

Demo : Area.

1. Customer
2. Administrator

Testing

* Reduce Errors , Runtime
* Bug free Applications
* Application is working as expected ( Client)
* Find out whether new changes are breaking existing changes or not.

1. Unit Testing

* Form of Testing in which individual components are isolated from the rest of application so that their behaviour can

Be validated

Functions /Methods

* Unit Test helps to test every small unit of source code.
* Mock the dependency of the code so that the individual unit of code can be tested separately.
* Unit test case do not detect issues in the interaction between components.

Create: Insert

Dep: repository

Dep : \_logger

Moq Object

Test Code

How many Test cases we should write

1. Ideally it should be equal number of methods
2. If we have conditional statements in these method, one test case for every conditional stmt

Index()

{

If() …logic .. Test

Else … logic … test

}

1. Unit Test cases, responsibility of Developer

Controller

Component

Model

Views

How to Write Testing Code

Technology

1. MS Test
2. NUnit
3. XUnit

* Free , open soure unit testing tool for .Net Framework.
* Written by Inventor NUnit
* XUnit works better with different tools like Resharper, code rush etc
* XUnit is available as Nugets

Test Case : Method

1. Arrange : Setting up the Object or Conditions for the test case.
2. Act : Performing the Test case
3. Assert : Verifying the result.

* Static class

1. Equal
2. NotEqual
3. True
4. False
5. isType
6. InRange etc

Controllers

* Central part of ASP.NEt Core application
* Data access logic should not be tested.
* Filters, Routing , model binding

Demo : testing controllers :

Components

1. Integration Testing