Current Trends in the IT Industry

* Linux Adoption : Reduce the Cost.
* JS Frameworks: Performance
* Microservice Architecture: Independently developed and deployed.

Assign more resources to specific feature.

* Cloud : Dynamic Scalability.
* Containerization : Docker Or K8 : Reduces the Cost , by reducing the amount of resources required.

Microsoft .Net : 20

* .Net 1.0..1.1 : Console App
* .Net 2.0 : Win Forms, WebForms
* .Net 3.0
* .Net 3.5 : Linq
* .Net 4.0
* .Net 4.5 : Async .. Await

Microsoft Rearchitected :

* .Net Core 1.0
* .Net Core 2.0
* .Net Core 3.1 == This
* .Net Core 5.0 : VB and Support EF6 removed

What is .Net Core

* Framework to Build Cross Platform Applications
* Cloud Ready
* Integrated with JS frameworks
* Easily create MS and Containerize

What Languages Support

* C#, F#, C++ as of 3.1

Application we can Create

* Windows Forms
* WPF
* Console Application
* Web Application
* gRPC
* Blazor = Angular Or React

How come .Net Core is Cross Platform

* JVM was present for every environment
* Core CLR Present for every environment

What is Requirements

* Visual Studio 2019 : Free
* SQL Server 2012 and Above OR SQL Express(Free)
* Postman : Free

Real World Application : Administrator to maintain Product inventory

[swpnlgkwd30/BoaSessionFour (github.com)](https://github.com/swpnlgkwd30/BoaSessionFour)

Lab1 : Create sample application

Project Structure

* Program.cs : Entry Point of your application.
* Startup.cs : Configure your application.

Configure : Responsible for Handling Request and response . It contains Middlewares.

ConfigureService : Responsible for Creating Service Objects. DB Objects, File Objects , Security Object .

* appSetting.json : Settings of your application.
* launchSettings.json : Modify port number or application url Or set up environmental variable.

Startup.cs

Program.cs

EF

Objects : DB, File

Model

1. Configure Service

2. Configure

Req

Browser

Ctrl

View

.cshtml

UseEndpoint

Routing

Middleware

Middleware

Razor Syntax : @

MVC

Pipeline

ASP.Net Core Platform

Middleware

- Section of Code that Modifies the Request and Response.

- Middleware is software that's assembled into an app pipeline to handle requests and responses.

- Middleware takes Request Delegate as Parameter

- Takes 2 Parameter

I .Context : Current Request and Response Object

ii. Next : Next middleware to call

Use : Takes Request Delegate as a parameter. Context , next .. if you don’t use next, next middleware will

Not execute . This is called as Short Circuiting the Request

Run : Use to Short circuit the Request. IT doestnot have next parameter

Types of Middleware

Built In : MS Provided

* DeveloperExceptionPage
* Routing
* UseEndpoint
* UseStaticFiles

Custom : You have to create your own

* Our own Middleware

Problem

* I have some CSS, JS , HTML in my project . I want to make it available to Browser whenever Requested

Solution : Whenever you want to make available Such Static files to the browser. You need to use Middleware : UseStatic

Static files, such as HTML, CSS, images, and JavaScript, are assets an ASP.NET Core app serves directly to clients by default.

Problem : wwwroot : Content from this folder is available without any auth . Important Static docs cant be stored here.

Solution : Create your own folder and Keep those docs inside this. Expose this folder from Pipeline and configure it to make available only after authentication.

Web application

* ASP

:- Runtime Error

: = Interpreted language

* ASP.Net Web Forms

Adv

* Faster Development
* Rich controls
* Compiled

Problem

* Testable
* Tightly coupled
* Don’t have control over HTML its going to produce.

ASP.Net MVC 1.0

* MVC Convention based not Configuration based.

Action Name

Controller Name

Model

Request : http://localhost:1385/Home/Index

Controller

Views

UI

View

Model : Holds Data and business logic

View : User interface logic

Controllers : handle the request.

Adv

1. Separation of Concern
2. Unit testable
3. Maintainable

2.0

3.0

4.0

MVC 5.0 = Develop and Deployed only on Windows

Problem

* Doesn’t have Rich Server Controls

ASP.Net MVC Core

* Build Cross Platform Web Applications.

Conventions

1. Controller should be present in Controllers folder
2. UI logic should be present in Views Folders
3. Every Controller name has to end with Controller word.

Problem : Home Page : Home, About Us and Contact Us. How to do this in MVC

Solution

* Home Controller

1. 3 Action Methods

Controller

* Handle The Requests
* Connect the model if required and pass data to View.
* Inherited with Controller class

:- It gives the methods required to return different types . View, Component, Partial View,JSON.

IActionResult

* Base Type for the all the other types
* Json, View , PartialView

Configuring Services

* Configure the Services.
* How many such object created is depend on the lifetime used.

Lifetime

1. AddTransient :

* Within a Request whenever Its resolved . One object created.
* Small Calculation logic

RandomService Object

HomeCtrl

Request 1

RandomService Object

RandomWrapperService

1. AddScoped : Per Request one Object is Created . DB Object or File Object

HomeCtrl

Random Service Object

RandomSWrapper

Request1

Random Service Object

Request2 ..n

1. AddSingleton : All Request one object is Created . Sharing Some Db .. in memory db

Problem : Pass Data from Controller to View

1. View Data Or ViewBag

* Problem

1. Doesn’t give Intellisense
2. If you make typo , you might lose data on screen

Drop down : selectedCategory = Electronics

1. Model

* Intellisense
* Reduce the Errors

Application : CRUD Operations Product Admin.

1. Controller : Home Controller --Done
2. Model :- Product Model --Done
3. Services that can perform this operation.
4. Consistent Look and Feel for all the Pages. -- Done

Problem : Consistent look and feel for all the Pages

Solution : Layout : MasterPage

Services : How to Create

* AddProduct
* DeleteProduct
* UpdateProduct
* GetProduct
* GetProductByID

1. Action Method
2. Action Method will call Service

HomeController

ProductInMemory

IStoreRepository

Service : Done

Problem : Craete a Link => Page that will display options to add new Product

Or Forms . We are not suppose to use HTML code directly

* <a href="Home/Create"> Add Product</a> : Not a good way

ASP.Net MVC 5

1. HTML Helper classes : Well integrated with ASP.Net MVC Framework

@Html.ActionLink("Add Product","Create","Home")

@Html.TexxtboxFor("Add Product","Create","Home")

@Html.Checkbox("Add Product","Create","Home")

Problem

* Anyone who is not C# guy , it will difficult to understand
* Design Web Page => HTML

MVC Core

1. TagHelpers

* Syntax is more like HTML. Designer can use & write it.
* Use to generate HTML elements on View
* Its also well integrated with ASP.NET MVC Framework.

1. Add Package for TagHelpers
2. Enable TagHelper for your application.

Random Service Object

Request1

Request2 ..n

Lab2 : How to Create Services and Configure the lifetime of the Same.

* Create RandomService : Going to Generate Random Nos

i. IRandomService --

ii. Implement --

* Random Service = > Controller Class

Singleton

1. Dependency Inject

Scoped

Transient

Random

HomeController

DI Syntax

IRandom

Service : Logic (DB)

Why it should be implemented as Interface .

* Loosely coupled.

OracleDatabaseService

GetData :

IDBInterface

GetData

Ctrl

IDBInterface I ;

i.GetData();

SQL

MySQLDtabaseService

-GetData

MySQL

Request => HomeCtrl => I Product :Oracle DB Service

Request => CustomerCtrl => Customer : MySQLDB Service

Empty Project

* launchSet,prope,appsetting, etc

gRPC : Web Service

Services.AddgRPC : Classses

Blazor = MS Version of Angular

Services.AddBlazor : Component

ASP.NEt Core Web App : services.AddController

Pass Data from View to Controller

1. Asp-route : Add Additonal Parameters to route and pass the data
2. Submit Action to Pass Data

Problem

* Number of Products are really huge and doesn’t fit into a single Page

Solution

* Pagination
* View having Links

1. Razor
2. TagHelpers

Custom TagHelpers

* Create a TagHelper which will work as Pagination.

1. Need a model that can store the information about the Current Page
2. Rewrite the query which can retrive the data based on pageSize

ViewModels

* Whenever we want to have multiple models within a single view . We should create viewModels.
* Class that represent the model of view

Routing Feature : http://localhost/Product/Page3

http:// localhost/Chess : Give me all records related to chess

<https://localhost/Cricket> : Give me all record related to Cricket

<http://localhost/chess/Page1>

Problem : User Should be able to navigation to these links

Solution : Create Links on the Left Section so that user can select particular categories.

Create Numbers of Categories and Generate the Links.

View Component

* Similar to Partial views
* Render chunk
* Business logic
* Invoked from layout page

<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/view-components?view=aspnetcore-5.0>

Real world Situation

1. Dynamic Nav Menu etc

Problem : Project is Ready and deployed. We get complaint from the Client that something is not working.

Solution: Logging .

How

* ASP.Net core supports logging but doesnot support logging in files.

Third Party logging providers

* NLog : <https://github.com/nlog/nlog/wiki>

: NLog is a flexible and free logging platform for various .NET platforms

* Serilog

Steps to use NLog

1. Download the library
2. Create Config file
3. Enable the Nlog for application
4. DI
5. Log it

Problem : Interact with FS

Solution : File Providers

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/file-providers?view=aspnetcore-5.0#:~:text=the%20directory%20folder.-,ASP.NET%20Core%20abstracts%20file%20system%20access%20through%20the%20use,Providers%20to%20locate%20static%20files>.

ASP.NEt Abstracts file system through File Providers.

Internally , Webhosting Environment and static files use File Providers.

PhysicalFileProviders: Wraps System.IO files and provide access to Physical file system

IFileProvider

File System

Application

System.IO

Routing

* Responsible for matching HTTP Request and dispatching it to the executable endpoints.
* Endpoints are the apps unit of executable Request handling code.
* Endpoints can also extract the values from the Request URL.

Middlewares

1. Router : app.UseRouting

* It looks for the set of defined endpoints and select best match based on the request.

1. Endpoints : app.UseEndpoints

* Configuration of Endpoints

Home

Index : Pageno

https://local/Customer/index/1

UseEndpoints

Routing

Create

Customer

ASP.Net MVC 5 :Previous : Separate file use to exist : routeconfig.cs : Routing related logic

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

Types of Routing

1. Conventional

* Routing logic in the Startup.cs.
* Endpoints

1. MapGet : Route that is not associated with any of the controllers. Execute some logic based on the route.

endpoints.MapGet("/authorize/{username}", async context =>

{

// read this route price

var userName = context.Request.RouteValues["username"];

await context.Response.WriteAsync("Hello : " + userName);

});

endpoints.MapGet("/productinfo/{price}", async context =>

{

// read this route price

var price = context.Request.RouteValues["price"];

await context.Response.WriteAsync("Hello : " + price);

});

1. MapControllerToRoute : Request Navigate to Specific controller.

endpoints.MapControllerRoute("catpage", "{category}/Page{productPage}",

new { controller = "Home", action = "Index", productPage = 1 });

// Route : http://localhost:5000/Cricket

endpoints.MapControllerRoute("cateogry", "{category}",

new { controller = "Home", action = "Index", productPage = 1 });

//Product/Page3

endpoints.MapControllerRoute("pagination", "Product/Page{productPage}",

new { controller = "Home", action = "Index" });

1. MapControllerToDefaultRoute : Request navigated to Default Route. Controller : Home, action =-index

<http://localhost/customer/index>

* First Part : Controller
* Second Part : Action

If we don’t pass anything by default values will be : Home and Index

1. MapAreaControllerToRoute = Application into Area.

Routing Constraints

* Add a validation into the Route Parameters

endpoints.MapGet("/authorize/{username:minlength(4)}", async context =>

{

// read this route price

var userName = context.Request.RouteValues["username"];

await context.Response.WriteAsync("Hello : " + userName);

});

endpoints.MapGet("/productinfo/{price:int}", async context =>

{

// read this route price

var price = context.Request.RouteValues["price"];

await context.Response.WriteAsync("Hello : " + price);

});

// Route /Chess/Page2

endpoints.MapControllerRoute("catpage", "{category}/Page{productPage:int:min(1)}",

new { controller = "Home", action = "Index", productPage = 1 });

1. Attribute : ASP.NET MVC5

* Routing logic in the specific files.
* Route attribute to specify route to specific action.
* Primary Adv

1. It allows you to define your routes in the same file as a controller so it make sure your startup.cs is more clean.

[Route("Account")]

public class AccountController : Controller

{

[Route("Login/{username:minlength(4)}/{password:minlength(4)}")]

public IActionResult Login(string userName,string password)

{

return Content("Login Called");

}

Problem

* Able to add Empty product into my DB

Solution

* Model Validation

Validation of ASP.Net MVC5 and MVC Core = Same

* Data Annotations

ViewModel

Create

Name of Product

Valid

Model : Product

Valid

Update

Product Name

Index

ProductID

Views tightly attached to model. So if anything changed in the model, every view which is using it

Will be impacted.

One View: Name : Price

Second View : Name :Category

By default this is Server side validation.

Remote Attribute

* Implements Client Side Validation that requires calling a method from the server side.
* Expects JSON response.

1. True : data is valid
2. False : Data is invalid

Real World Example

* User enters username in textbox, we have to validate whether username is already taken or not.

Logic

swapnil

Json : true Or false

Category : Cricket , Chess and Soccer : ServerSide Logic which will validate this.

AutoMapper

* Problem : As a Best Practice when we introduce ViewModels in our project we have to convert VM=> M and Model=>ViewModel

Solution : AutoMapper

* Package the Converts One object to another object.
* AutoMapper is a popular object-to-object mapping library

1. Install the Package—Automapper and DI
2. Create a class that should be inherited from Profile class
3. Constructor write a Mapping Logic
4. Use IMapper interface
5. Configure Service to find Mapping logic in the current Assembly.
6. Use Map Method.

Developer Exception Page

* Contains lot of sensitive information about your code. So it should be available only during development and not on Production.

Solution

* It should Create Custom Error Page.

: Controller

: Index View

Status Code

* An Additional method of handling errors.
* UseStatusCodePages Middleware
* Use to Capture html status codes between 400 to 599
* Display short explanation of error code to the user.

Areas

Problem

* Naukri.com : Application is used by 3 Users

1. Employer : Area

* Set of Function
* 10 Controllers \* Models , \*ViewModels

1. JobSeeker : Area

* Set of functions
* 20 Controllers

1. Administrator : Area

* Set of function
* 30 Controllers
* ASP.NEt MVC5 == ASP.NEt Core
* Organize Related functionality into Group as a Separate.
* Partition an ASP.Net Core Web app into small functional group

Functional Group

1. Pages
2. Controllers
3. Views
4. Master Pages
5. Application
6. Developer : Linq
7. User

When to go for Areas

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

Problem

1. We have two users i. Admin ii. Customer

Solution

* Areas

I .Admin

1. Customer

Problem

* Data is Coming from the in Memory Collection.

Solution

* Connect to Database.

Entity Framework.

Problem : Data Access Code before EF : ADO.Net : cmd. commandExcuteQuery(“select \* from Product”)

* Lot of Runtime Errors
* No Rich intellisense.
* Lot of Code.

Object Relational Mapping : EF

Shopping

Sql Server : Relational

ShoppingContext

* ConnectionString

DbSet<Product>PRoducts

Objects

Tracks

Products

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Cost | Category |
|  |  |  |  |

C# Application

Class Product

ID,

Name,

Cost,

Category

EDMX

Table = Class

Column = Properties

Database : DataContext

Advantage

1. Developer have to write less code.
2. Intellisense
3. Less Run time Errors

What is EF

* Object Relational Mapping Framework use to Connect to Database.

Different Versions

1. 5.0 : .NET Framework
2. 6.0 : .Net Framework

EF Core

* Lightweight, Open Source and Cross Platform Version of EF.
* EF Core 3.1 runs on .NET Core and .NET Framework, through the use of .NET Standard 2.0. However, EF Core 5.0 does not run on .NET Framework.

1. .Net Framework Application can use EF Core 3.1
2. Application that can have both EF6 and EF Core 3.1
3. ASP.Net Core cant use EF6.0 after 6.3 you can run EF on ASP.Net Core

<https://docs.microsoft.com/en-us/ef/efcore-and-ef6/>

Compare EF6 with EF Core

1. EF Core does not have Graphical Visualization
2. Ef6 Supports EDMX file that is not available with EF Core.

How to Integrated EF Core in ASP.Net Core Application

1. Install libraries :

* EFCore
* EF.Design
* EF.SqlServer

1. Build Context Class :
2. Configure and Read ConnectionString

* IConfiguration

1. Enable Context and Pass the ConnectionString from Startup.cs
2. Data Access Code :
3. Change the Ref to enable SQL Repo
4. Migrations : Generate necessary tables and Database Required by your application.

PM> dotnet ef migrations add Initial

No project was found. Change the current working directory or use the --project option.

PM> dotnet ef migrations add Initial --project sample-app

Build started...

Build succeeded.

Done. To undo this action, use 'ef migrations remove'

PM> dotnet ef database update --project sample-app

Build started...

Build succeeded.

Problem Whenever any sales happens the Inventory Quantity should be reduced by 1. Both operations should be atomic.

* Sales
* Inventory

Transaction support of EF Core.

* Commit, Rollback.

Problem

* Existing .Net Application : EF 6
* Performance : EF Core

Solution :

* .Net Framework => EF Core

Database first approach for EF Core.

* Scaffold-DbContext "Server=Swapnil-PC\SQLEXPRESS;Database=BoaSessionFourSat;Trusted\_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer

Web API

Need

* When 2 application wants to communicate with each other on Internet.

App2

App1

Solution

1. Remoting : .Net => .Net , Binary , Fastest
2. Web Services : .Net => Java, SOAP , Slow
3. WCF : SOAP,HTTP,Faster . Problem : Lot OF Configuration
4. Web API : HTTP based Service, Format : JSON
5. GRPC : Performance wise 7..10 times better than Web API.

Rest Service

* Standard for interactive applications that uses Web Services
* Provides web resources in a textual representation to allow them to read and modified with a

Stateless protocol and predefined operations.

Web API

* Accept the Request and Generate Responses that contain data
* Provide access to application data to Client side applications (Angular, React Ember etc)

<http://localhost/product/Getproduct>

HTTPMethod + HTTPVerb

Application 2

* Web API

GetProduct

AddProduct

DeleteProduct

UpdateProduct

Application 1

<http://localhost/product/Getproduct>

-Angular , React,ASP.Net Core Web

HTTPVERB: GET PUT,POST ,DELETE

JSON format + Status Code

Project Structure is same as ASP.Net Core

1. Controller : ControllerBase

* Access to features from MVC framework

I .HttpContext,ModelState,Request,Response,RouteData and User

* Makes Sure you cant return view from APIS.

1. Controller Attributes

* Route : [Route("[controller]")] = Enabling Attribute Routing
* [HttpGet] HttpPOST,HTTPDELETE, HTTPPUT

1. Binding Source
2. FromBody : Request Body
3. FromQuery : QueryString
4. FromRoute: Route Parameter
5. FromHeader : Header Information

How to Connect to Database and Get the Data for Web API.

* Libraries --
* Context --
* ConnectionString and Read it
* Enable the Context in startup.cs
* Use Context in Controller

Testing APIS

1. Browser
2. Postman : Useful During Development API Testing
3. Swashbuckle: GUI that you can add for Testing your api.

Best Practice

1. Create Asynchronous Action Methods : Wherever Async method is der use it.

User1

R1 : T2,

:.. 1 min

R2 :.. 2 min

R3 : T3 : 3 min

R4 : T1

1. Pass Valid Status codes to Client Application

Let say Get Product that return null in this case we need to send Not Found that is 204.

1. Ok : 200 : optional Object in the response body
2. BadRequest: 400 model state error
3. NotFound: 404
4. Validate Data through Data annotations
5. Project Selected Properties
6. Use Swashbuckle : Open API Specification that will automatically generate description of web Service.
7. Install the Library

dotnet add package Swashbuckle.AspNetCore --version 5.0.0-rc2

1. Enable Swashbuckle for your application.
2. Content Formatting

* Web API always returns a data in JSON format. But web api is capable of sending multiple format based on the configuration.
* Content Format Selected based on

1. Format the client accept.
2. Format that application can produce.
3. Content Policy Set by the action method.

Content Negotiation

* Most clients include an Accept header in a request which specifies the set of formats that they are willing to receive in the response.

Respecting Headers

* Whenever you asked for a format that doesn’t supported by the application. Web API will always give you JSON format.
* Ideally in this case application should get 406 error that is format is not available

1. Adding Produces and Consume Type

Type : GET/PUT

URL : Where

Success : function

Failure : function

datatype:json

Consuming Web API

App

C#, Angular , React

Web API

GetProducts

GetProductId

HTTP Library

* Http
* Axios
* Ajax
* HttpClient
* jQuery => Ajax : Web API Call

Enable Cors

* Browser security prevents a web page from making requests to a different domain than the one that served the web page. This restriction is called the *same-origin policy*. The same-origin policy prevents a malicious site from reading sensitive data from another site

ASP.Net Core Identity

* What is Identity
* Enable Identity
* Login,Logout, Register --
* Json Authorization
* AAD
* Membership system for web application
* API the supports login, logout and Register functionality.
* Manage user , passwords , profile data
* Identity can be configure to use SQL server DB or Azure TableStorage.

Microsoft.AspNetCore.Identity.EFCore

SignInManager :Login and Logout

UserManager : Managing users.

IDentityUser : Represent user

IdentityRole : Represent Role

SQL

Identity Tables

Application

Login

Logout

Register

User

* User is Represented using IdentityUser class.

How to Enable Identity

1. Install the library : Identity.EfCore
2. Create User class : IdentityUser
3. DataContext class should be inherited from IdentityContext.
4. Enable Identity in Startup.cs
5. Lets create default roles for the user.

Registration

* Controller : AccountController :--
* Views : Register :--
* UserManager

Login and Logout

* Controller : AccountController
* Views :
* SignInManager

Problem : Allow Access to user only if he is logged in.

Solution : Filter : Authorize

Problem : Login, Logout and Register all 3 options are always available on our screen.

Solution :

1. Layout hide the links based on the users logged in status.

Problem : How to Authorize Web APIs . I want to give access of our Web API to valid requests.

Solution : JWT token.

Json Web Token

Web API

Login

GetProducts

AddProduct

Application

JSON Web Token

Payload : User Related Info

Header: Type of Token , Algo

Signature:Digital Sign Generated by combining header and payload

Json Web Token

How to Enable JSON Web token Auth

1. Install the libraries : Identity and JWT libraries.

Microsoft.AspNetCore.Identity.EFCore

Microsoft.AspNetCore.Authentication.JwtBearer

1. Enabling Identity for Authentication

* User class
* Configured Context to use Identity
* Enabled identity in startup

1. Enable the JWT Authorization for your application.
2. Issuer : Actual Server who created the token
3. Audience : Receiver
4. Lifetime
5. Key : Secret key

Microsoft.Identity

Microsoft.Identity.UI

Azure AD

ClientId

User Information

Email ID

Password

Boa Application

appSettings

Token

Microsoft Page

Application :

Boa : Client ID

1. Add Package

* Microsoft.Identity.Web : Main Package
* Microsoft.Identity.Web.UI : Add the UI for user sign in and signout

1. Enable AAD Authentication From startup.cs.

Filters

* Filter inject logic into Request processing.
* Allows us to run certain logic before or after specific stages in the request pipeline.

Need

* Avoid Duplication of Code
* Error Handling
* Logging
* Caching

Real World Example

* Uploading some file and want to find our the current status .

How its different from middlewares

1. Middleware cant access the ActionExecutionContext , rather it has access to HttpContext
2. Middleware run on every request regardless of controller or actions

Filter Types

1. Authorization

* Built in Filter

1. Resource Filter

* Use for Caching

1. Action Filter

* Use to modify the request before its received by an action method .
* Use to Remove the duplicated code.
* ActionExecuting
* ActionExecuted

1. Exception Filter

* Handle unhandled Exceptions
* Doenst have before or After Actions
* Implements OnException Method
* Method will be called whenever any unhandled exception occurs

1. Result filter

* Use to alter the action results
* Use to alter the result produced endpoint
* Generally last minute changes done through this filter.

For Example : Adding some Response header information.

Difference between Filter in MVC 5 and MVC Core

* Filters can be configured as Scoped,Transient Or Singleton.

Problem : Whenever user Enter Price of the Product ==0 , I want to navigate him to some error page which says. “Price =0 is not acceptable”

Solution : Exception Filters

How to create

1. Create class : IException
2. OnException Method Implementation
3. Custom Error Page that takes this Exception and print it.
4. Activate this Custom Exception.

Problem : User is going to Pass Culture code from the URL , we want to read that culture code and convert it into Culture information

http://localhost/?culture= fr-FR => Hello : France.

Solution

* Middleware : Custom Middleware

http://local/?culture=fr-FR

Pass the Culture information which will print this information

Middleware : logic

1. Create class
2. RequestDelegate as Parameter in ctor that point to next middleware
3. Implement InvokeAsync method that contains the logic and will automatically gets called.

Cross Site Request Forgery

* Cross-site request forgery (also known as XSRF or CSRF) is an attack against web-hosted apps whereby a malicious web app can influence the interaction between a client browser and a web app that trusts that browser. These attacks are possible because web browsers send some types of authentication tokens automatically with every request to a website. This form of exploit is also known as a *one-click attack* or *session riding*

1. User Signs to hdfcbank.com .. Cookie on the client side (Browser)
2. User Visits malicious site. Bad.com

<h1>Congratulations! You're a Winner!</h1>

<form action="http://hdfccom/api/account" method="post">

<input type="hidden" name="Transaction" value="withdraw">

<input type="hidden" name="Amount" value="1000000">

<input type="submit" value="Click to collect your prize!">

</form>

1. When the user click on submit button . Authentication cookie will be use <input type="submit" value="Click to collect your prize!" Request
2. The request runs on the www.hdfcbank.com server with the user's authentication context

Solution

* Anti Forgery tokens comes into the picture.
* Whenever generally forms are generated by .Net Core . To every form ,they add some tokens. So now We have 2 tokens

1. Authentication cookie
2. Forgery tokens which is part of form , as a hidden field.

* Whenever request comes, Server verifies both of these tokens.

Cross Site Scripting

* Enables an attacker to place client side scripts into web pages.
* When user load this pages the attacker script runs
* Form (No Validation) : Textbox : <script> nice logic </script> => submit . Script stored in the database
* Any Other user if he load this data from the database : Script : run by the browser.
* Querystring : I will pass script into QueryString

1. Validation.
2. Encoding : Any text between browser and Server should be encoded.

Space : %20 < : ;lt > :;gt : Use for Scripts

@Html.Raw : Disable the Default behaviour and script will start executing.

Blazor

* Browser + Razor
* Single Page Applications.
* Alternative for Angular and React

Traditionally :Server Side

Server

Browser

Home AboutUS CU

Problem

1. User Experience and Performance
2. Every Request user has to wait and application becomes irresponsive.
3. Lot of load on the Server

Modern :Single Page app : We have single page where all other pages will be loaded as partial pages.

Server

Browser

Home AboutUS CU

Brings all contact

App

Web API

App

Data

Adv

1. User Exp and Performance
2. Responsive
3. Reduce the load on server

Problem

1. Initial load may be slow

How to build SPA

1. Angular = Google : TypeScript , Angular Framework
2. React : Facebook : JS : Javascript, React
3. Ember : JS
4. Vue JS
5. Blazor : Microsoft : C#

What is Blazor

* Create Rich interactive UI using C# instead of JS.

Front End

C#

Back End

C#

* Both Client and Server can Share logic .
* Well integrated with modern hosting platform like docker.

When to choose Blazor over Angular or React

* Depend on Team Exp : if No JS exp then go for Blazor.
* Else you to learn typescript and use Angular or React.
* Accept blazor with all its inefficiencies.

How Writing code in .Net will help

* Leverage the Existing Library
* Share the app logic between client and Server

Where Blazor can run

* Safari
* Chrome
* Edge
* Firefox

Tool

* VS Code
* Visual Studio

Blazor Hosting Models

1. Client side Hosting Model

* App is Directly executed in browser.
* .Net Runtime + App + Dependencies downloaded in browser.

Browser

DOM

Blazor.webassembly.js

Server

.Net Standard (DLL)

Blazor Framework(DLL)

C#

Blazor app(DLL)

Adv

1. The app is fully functionally after its downloaded to the client.
2. Client Resource leveraged
3. Work is Offloaded from the Server.

Disadv

1. The app is restricted the browser capability.
2. Web Assembly support is required
3. Download size is large hence initial load takes time.
4. Server Side Hosting Model

* Application runs on the Server not on the client side
* Only changes to the DOM is transferred to the client.
* Constant Connection is required between client and Server

Server

.Net Standard Or Framework

Blazor Framework

Browser

SignalR :Persistent

DOM

Blazor Application

Tree

Blazor.server.js

Adv

1. Download Size is Smaller so Initial load much faster
2. App takes full capability of Server.

Disadv

1. Higher latency
2. No Offline Support
3. Scalability
4. Client should connect to same server if the connection is lost.

Use Case

1. When WebAssembly not supported on browser where you want to run the application or Initial load much faster.

Project Structure

1. Program.cs
2. Startup.cs
3. App.Razor : Routing logic
4. Import.Razor : Location where all the required namespaces are stored which will be used by all the views.
5. Shared : Contains Master Pages
6. Pages : Components and Single Page

* \_Host : single Page application
* Components

Components

* Building Block of Razor Application
* .razor Extension
* Razor components are delivered to the browser as part of application.
* Part of Page
* Component in Razor

1. @code : HTML Code
2. C# logic :
3. @page : Route . How you will reach this particular component.

Data Binding

* Each app receives input and display output.
* Process by which HTML from @Code is bound with logic.

Types of Binding

1. One Way

* Class Variable or Properties are bound to HTML for display

1. Two way

* Generally used with input controls with the help of @bind attribute.

@bind = “Field/Property/EventName”

Demo : Binding with Textbox, Checkbox, Select, Conditional Attributes

Event Handling

* HTML element attribute @on{Event} with a delegate Typed
* Passing any parameter than use Lambda expression in the delegate type.

<input type="text" **@bind**="ProductName" />

<button class="btn btn-warning" **@onclick**="()=> SearchProduct(ProductName)"> Search Product</button>

Communication between Components

Emp1

Textbox : EID

Emp2

GRID

Routing

C3

C1

Cascading Values

C5

EventCallback

Parameters

C2

C6

1. Parent To Child : Parameter

* Child Content : Components can set the content of another component

Real World : AlertComponent , That show the Message

1. Child to Parent : EventCallback<Product>
2. Parent – GrandChild : Cascading Values
3. Parent to another Parent : Routes

* NavigationManager is responsible for navigating from one component to another.

How we can inject Object or Properties or Dependencies we required, its equivalent to Ctor injection syntax

public HomeController(IRandomService randomService, IRandomWrapper randomWrapper,

IStoreRepository repository, ILogger<HomeController> logger, IFileProvider fileProvider, IMapper mapper,UrlEncoder urlEncoder)

{

\_randomService = randomService;

\_randomWrapper = randomWrapper;

\_repository = repository;

\_logger = logger;

\_fileProvider = fileProvider;

\_mapper = mapper;

\_urlEncoder = urlEncoder;

}

@inject NavigationManager NavigationManager

Best Practice

Key

* While rendering list of Elements , blazor diff algorithm , it must decide which of the previous elements must be retained.
* It works fine. But when collection content change due to some insertion, deletion or re-ordering this might causes complex re-rendering
* This can be controller through Key.

Attribute Splatting

* Useful when defining component that produces a markup element thar support variety customization.
* Attribute can be splatted on to Element.

Templated Component

* Generic Component.
* Generally used for List and Tables

CRUD Operations in Blazor

1. Display all Data from Database in application

Add a New Product

Blazor Form Components == TagHelper

* To Create Forms , Blazor has given some built in components to render form elements
* Ensures that Server side properties are updated after user interaction and integrating validation.
* Built In Components

1. EditForm
2. InputCheckBox
3. InputDate
4. InputNumber
5. InputTextArea

Component to Component Communication

1. P=>C : Parameter
2. C=>P : EventCallback
3. C=C : Routing
4. P=>G : Cascading
5. P<=>C : Chained Binding
6. P=>Call Function=> Child = Ref

**Chain Binding**

* Two way data binding between Parent and child component.
* It allows us to update the child component values from the Parent component and vice versa.
* In order to do that we need to use @bind-{Child Property}
* We need to create EventCallback to update the value Parent

**Ref**

Parent

Child Component

Add

Delete

Update

<Child @ref=”child” />

C#

ChildComponent child;

Child.Add();

Child.Delete()

**Blazor Component Library**

* Reusable Components

**Lifecycle Methods in Blazor**

* within different stage of component loading to destruction if we want to write a code.

1. SetParameter Or Async

* First Method invoked
* Called before any parameter value is assigned to their respective properties.
* Called each time a new or updated parameter received

1. OnInit or Async

* When component is loaded.Writing call to DB logic

1. OnParameterSet Or Async

* All Parameters have been assigned to their respective properties.

1. OnAfterRender Or Async

* This methods is essential if you want to perform additional initialization steps like loading some custom js libraries or third party js libraries.
* Activate Third party libraries so that it can operate on rendered DOM elements.

1. ShouldRender

* Decide whether UI should Re-rendered or not
* Use to suppress further re-rendering of UI.
* It returns true or false.

State Management

https://docs.microsoft.com/en-us/aspnet/core/blazor/state-management?view=aspnetcore-5.0&pivots=server

Security

<https://docs.microsoft.com/en-us/aspnet/core/blazor/security/webassembly/?view=aspnetcore-5.0>

GRPC

* What is gRPC
* What is Proto
* Web API vs Grpc
* How to write code in gRPC

Application communication

1. Remoting
2. Web Service
3. WCF
4. Web API
5. gRPC

What is gRPC

* Open Source RPC Technology
* Developed by google in 2015
* Roughly 7..10 times faster than Web API.
* Supports Bi-Directional Communcation, enterprise application or streaming application (You tube)

Benefit

1. Performance.

Proto (Kind of WSDL from web service)

* Way to Serialize data , much more simpler , compact and faster
* Proto language which is supported by C#, Python, Go, JAVA
* Smaller Serialized as binary strings

gRPC Client

* grPC.Net.Client
* Google.Protobuf
* gRPC.Tools : Generate classes

1. Unary
2. Server Streaming
3. Client Streaming
4. Bi-Directional Streaming

gRPC Server

GetProducts

AddProduct

DeleteProduct

1. Unary

* Client Send a Single Request and get a Single Response

1. Server Streaming

* Client send a request and server send Stream of Response

1. Client Streaming

* Client writes a sequence of messages and send them to the Server, while server returns a single response.

1. Bi-Directional

* Both sides send a sequence of messages. Chat application.

Project Structure

1. Startup
2. Program
3. appSetting
4. Services: Contains the logic of your Service methods
5. Protos : Contains Proto files

That also called as Interface description language.

1. Service: Contains what all methods it expose

// Sends a greeting

rpc SayHello (HelloRequest) returns (HelloReply);

rpc SayBye(EmptyModel) returns ();

1. Messages that this service method accepts and returns. The properties inside this messages should be Ordered.

message EmptyModel{

}

message ByeReply

{

string reply=1;

}

Problem

1. GetProducts, GetProductById

* Steps

1. Create ProductService --
2. Create Proto file --
3. Add necessary models and methods in proto files. –
4. Register your Service
5. Generate Server Side classes.

Web API vs Grpc : https://docs.microsoft.com/en-us/aspnet/core/grpc/comparison?view=aspnetcore-5.0

* Proto files
* Http2 : Really fast , It supports multiplexing . you can pass multiple request over single tcp connection. / HTTP
* Binary / JSON
* Client, Server and Bidirection vs Client Server
* No browser support (GrpcClient .. grpc.NEt.Client,grpc.Tools,protobuf) /Yes

gRPC consumed

* Its consumed as JSON from UI like angular by converting grpc response to JSON.
* Server side Technologies like C# or ASP.Net Core MVC , you can directly use gRPC client.

Microsoft working on exposing grpc as Web API. Preview

Migration

* Port .Net application => .Net Core
* Complexity of the project
* Project model is available or not. For example : WCF Service in .Net… Not Possible Alternative: grpc
* Unavailable Tech

I WCF

1. WF
2. Remoting
3. CAS
4. WCF

What is Possible

1. Winform
2. WPF
3. Asp.Net MVC5
4. Web API2 => Web API Core.
5. Console Applications.

Tools

1. .Net Analyzer : Tells us how much your code is migratable
2. Try-Convert : Converts application directly from .Net Framework to .Net Core.

etc

Scenario : Project model available and Structure is same

Migrate Windows Forms Application to Windows Forms on .Net Core

1. Back up your project
2. Analyze project for portability

* <https://docs.microsoft.com/en-us/dotnet/standard/analyzers/portability-analyzer>
* Use Portalbility analyzer downloaded from above link. Purpose of this Tool is to Check your code and tell you how much of your code can be converted into .net core. Create different files for showing that information. For example Remoting, Code Access Security
* Project Model is available and Project Structure same

Scenario 2 : Project Model Available but Structure is Different

Migrate Web API2 to Web API Core

* Back up project
* Analyze the project for portability
* Project model available or not = Yes but project structure is different.
* Migrate

How to do it.

* Web.Config => appSettings.json
* Global.asax : Configure
* Routing Logic : Route Attributes and Http Verbs
* Ref that is not available on .Net Core : Need to find alternative
* Copy and Paste the logic( Controllers , Models ) of Web API Framework
* Controllers in .Net Core : ControllerBase
* ReturnType of Action Method is IActionResult
* Add Attribute Routing and necessary attribute in controller file.

Scenario 3: Project model not available

* WCF
* ASP.Net Web Forms etc

Solution : Find alternative technology and Rewrite the logic for that Tech.

WCF => Web API Or gRPC

gRPC

* The experience of developing and consuming service with gRPC is similar to WCF.

Adv

* Performance
* InterOperability
* Streaming : Bi-Directional communication.
* gRPC is similar to WCF : because the client and server classes are generated.

WCF

* TraderSys :Soln Name

Project

1. Library : TraderSys.PortfolioData

* Model : Portfolio.cs
* Logic : IPortfolio and Portfolio

1. WCF App: TraderSys.Portfolios

* Portfolio.svc

gRPC

1. Create Solution Name : TraderSys --
2. Add Application : TraderSys.Portfolios --
3. Proto File : Interface Description language --
4. All Data Contract Convert into proto messages—
5. Convert the Service Contract to Service
6. Add Library

Try-Convert

* Tool that convert entire project or solution to .Net Core
* It may reject some of project based on compatibility.
* This tool is not recommended if your project has custom tasks or custom imports.
* This tool work only on windows.
* Class libraries, Console application, Scheduled tasks can be easily converted into .Net Core.

.Net Standard

* .Net Standard is Cross Platform Implementation of .Net Framework.
* .Net Standard is Set of APIS that is available on .Net Implementation.
* BCL library that supports wide range of technologies.
* Use standard library , when you to increase the number of applications support.
* Solves the code sharing problem.

.Net Core Code Execution

C#/F#/C++

.NET Compiler Platform, also known by its nickname Roslyn, is a set of open-source compilers

C#, F#, C++Compilers : Code Name: Roslyn

MSIL

CoreFX : Reimplementation of class libraries for .NEt Core

Core CLR/ RyuJIT : Code Name for .Net Core JIT Compiler

Native Code

Testing

Advantage of Testing

1. Test the logic
2. Addition of new features is not breaking existing logic.
3. Improves the quality of code.
4. Check whether product is behaving as per clients requirement.

* Unit Testing

: Form of Testing in which individual components are isolated from the rest of the application so the behaviour can be validated

: Helps to test small unit of source code

: Unit test case do not detect issues in the interaction between components , its purpose of Integration testing.

How many unit test cases we should write

* Ideally it should be equal to number of methods available in application
* If this method contains any conditional logic then we have to write one test case per condition.

Testing Controllers

* Controllers are central part of ASP.NET MVC Core application.
* Testing logic for controller should not focus on business logic like data access.

What all things needs to be tested

1. ModelState
2. Error Response
3. Appropriate ActionResult.

How to do this

* Solution : Unit Test Project

1. Create unit test project that holds your test cases.

Project Tools

1. MS Test
2. NUnit
3. XUnit

* Free , Open Source unit testing tool for .Net
* Written by inventor of Nunit
* Xunit works better with Resharper and code rush
* xUnit is available as nugets.

Test Case

* Name of method describes what it does.
* Fact : Indicates that it’s a test.
* Test body

AAA Pattern

1. Arrange

* Setting up conditions for the rest.

1. Act

* Perform the Test

1. Assert

* Verify the Test Result.
* Expected , Actual
* Static Methods
* Equal
* NotEqual
* True
* False
* ISnull
* ISRange etc.

Mocking

* Classes for which fake implementations are required , we can provide mocks.
* Libraries

1. NSubstiture
2. Moq

Integration Testing

* Verifies different part of application work correctly together.
* It involves testing application infra concerns like DB, filesystem, NW Resources.
* Testing the logic within your own method , usually comes in unit tests while how your application works with framework and database that comes in Integration testing.

Best Practice

* Limit the number of integration test.

Problem

1. We cant connect to dev database because every time integration test runs, it might pollute the data.
2. Startup.cs create entire infrastructure that needs to connect with the database. So now who create that infra for us integration test project.

Services.AddScoped

Service.Connectionstring

Solution

1. In Memory database : Install Microsoft.EF.InMemory
2. AspNetCore.MVC.Testing : This package provides WebApplicationFactory class that helps us to bootstrap our app in memory.
3. We need to create dependencies of app In memory with respect to all infra required. Add a class for that.
4. ConfigureWebHost
5. Create a class that contains code of Testing apis.
6. C# => API : HttpClient

* Integration Testing

What is Microservices :

* Divide your entire application into small small services. Related to your back end logic

Shopping

1. Order MS : Web API : SQL = Domain logic , domain model
2. Product Info : Web API : Mongodb : Domain logic , domain model
3. CheckOut : Web API : Redis Cache : Domain logic , domain model
4. Admin : Web API : Oracle : Domain logic , domain model

MS : concept , Implemented : Web API, Spring boot, grPC

Benefit

1. Independently scalable : Order Service Required more resources
2. Deployable
3. Agile
4. Lightweight
5. CI/CD

Disadvantage

1. Monitoring
2. Multiple deployments : for Dev Ops

MS1

Cleint

Gateway

MS2

Gateway : Istio/Api Management/Traffec/Ocelot

* Authentication, Authorization Http Offloading

High available

1. Cloud
2. Kubernetes

gRPC : 4 hr

Migrations :4 Hr

Unit Testing : 4hr