Info Log

Fatal Log

^^ Different Levels

If the app breaks, you can figure it out.

In azure, Analytics.

When it runs on the machine it logs on console, when it runs in azure it logs to SQL lite database.

Configuration

Logging

Analytics (Application Insights)

Code Analysis

Configure your system so that when you run it on machine in dev, it runs on console.

WHERE IS THIS CONFIG STORED ^^

Microsoft.Extensions.Configuration //Use this API for CONFIGURATION

new ConfigurationBuilder()

.SetBasePath(Direcotry.GetCurrentDirectory()).AddJsonFile("appsettings.json", true, true).AddJsonFile($"appsettings.(Environment.GetEnvironmentVariable("ASPNETCORE)ENVIRONMENT") ?? "Production").Json", true, true)

.addEnvironmentVariables().build();

Need 3 different providers for config data - one of them must be an environment variable. Doesn't matter order but need to comment explaining the why you did it in that order.

Theres a UI in azure where you can go configure environment variables. - you should use this so when you delpoy your app it just works in production.

Need to specify logging location in environment variable. In production ittl log to database (via azure env var UI), and in CLI on local machine ittl Log to Console.

MIcrosoft.Extensions.Logging (It provides you with the ability to have log messages)

/////

/// THIS CODE BELOW IS SKELETON FOR LOGGING

private IUSerService UserService {get;}

private IMapper Mapper {get;}

private ILogger Logger {get;}

public UsersController(IUserService userService, I Mapper mapper, I Logger<USersController> logger)

{

UserService = userSErvice;

Mapper = mapper;

Logger = logger;

}

// USING ABOVE CODE YOU CAN DO "Logger.WhateverCalls"

/////

DO ALL OF THIS FOR THE API PROJECT.

IF I WAS EVIL - All Logging should come from WEB Project, But appear sequentially with your API project.

You could fix this by implementing a new version of ILogger to have all of your logging route through the ILogger instance to have it access the DB where you will store your Logs.

Inside WebHost Builder - you can add ".UserSerilog();" (This happens after .UseStartup in Program.Cs under IWebHostBuilder. These can be plugged into your code Via Dependency Injection through Microsoft.Extenstion.Logger (thing we specified above).

Get Extension Method from Provider. Add to Code via Dependency Injection.

Give ScreenSave (Save .HTML) to APplication Insights, or a URL that Mark has Access too.

Code Analysis. (Warnings Turned on As errors).

Add RULE set so that Localization is not required

Add a globalsuppression.cs file to allow Underscores in names of Tests.

GlobalSuppresion is Project Specific, Rule set is Solution Specific

EC OPPURUNITY -- Configure your Rule set for entire solution for one file. Directory.Build.Props File that configures rule set for entire solution only once. Also look into Directory.Build.Target file.

How to put settings at a directory level for project for MSBUild file rather than .csproj file.

COOLEST EC -- One solution that only applies to everything, and another rule set that only applies to test-projects.

GUIDELINE - if you put an exception into a suppression file, you must provide a Justification.

Micorosft.AVC.CodeAnalyzers??