MONASH University

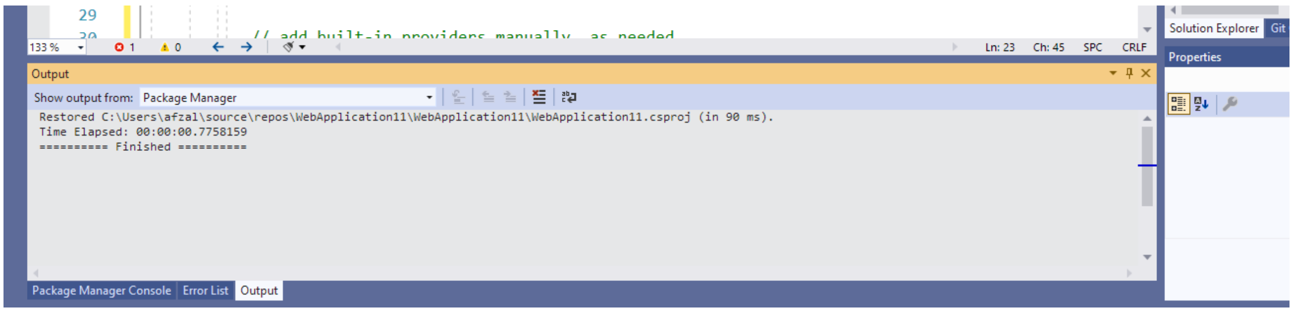
FIT5032 Logging in ASP.NET Core

Author: Afzal Thengakoottungal

POST-CLASS ACTIVITIES

Introduction

Logging is a necessary part of development. ASP.Net Core has built-in logging functionality that can be used. We can also use third party logging solutions such as SeriLog, NLog etc which are most robust and structured. In this activity, we look at the built-in functionality and a third party framework NLog. You can view all the logs in the Output tab of your Visual studio using the built-in logging functionality.



```
Restored C:\Users\afzal\source\repos\WebApplication11\WebApplication11\WebApplication11.csproj (in 90 ms).
Time Elapsed: 00:00:00.7758159
========== Finished ==========
```

If you cannot see the output tab, go to View and click on “Output”.

BUILT-IN LOGGING

Log Levels

6 log levels can be used –

1) Trace
2) Debug
3) Info
4) Warning
5) Error
6) Critical
7) None

The most basic log message involves a call to the ILogger.Log() function. We need to pass the log level and a text string to this function.

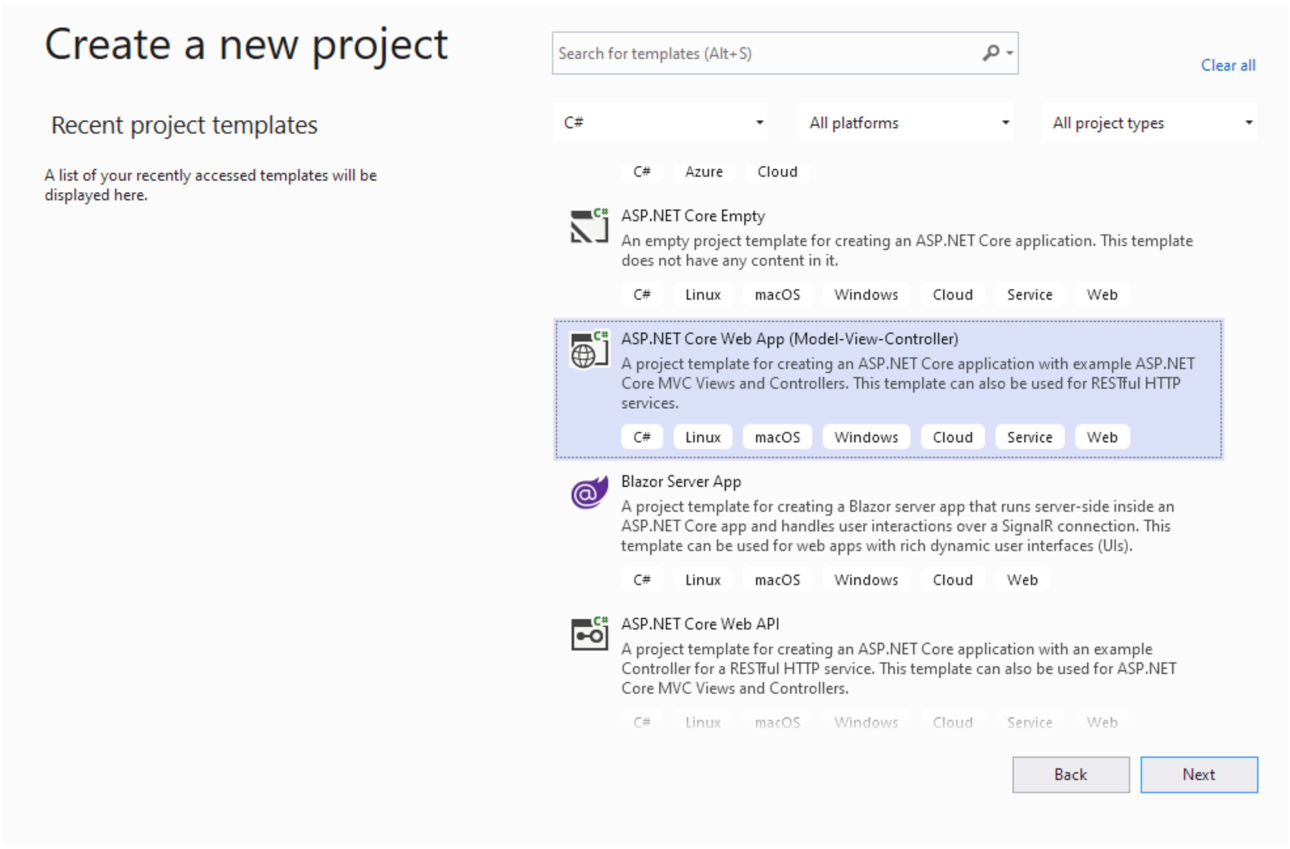
Eg: - _logger.Log(LogLevel.Information, "some text");

We can also use specific log methods such as LogInformation, LogError etc to log the message

Eg:- _logger.LogInformation("some text");

Step 1 -

Create a new web application based on .Net Core 5



Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

Search for templates (Alt+S)

Clear all

C# | All platforms | All project types

C# Azure Cloud

ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
C# Linux macOS Windows Cloud Service Web

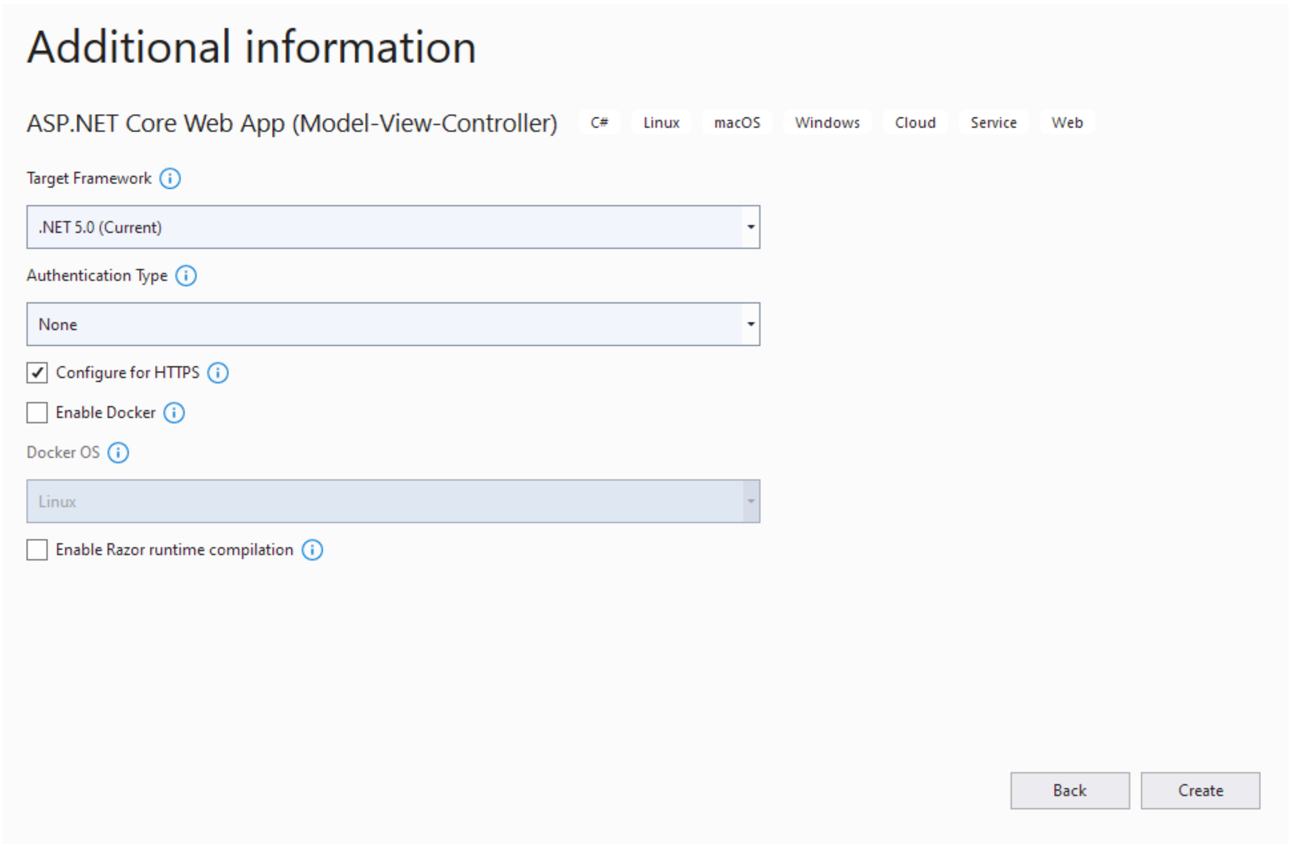
ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
C# Linux macOS Windows Cloud Service Web

Blazor Server App
A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web

ASP.NET Core Web API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
C# Linux macOS Windows Cloud Service Web

Back | Next

Provide the application name and select the target framework to be .NET 5.0 and click on create



Additional information

ASP.NET Core Web App (Model-View-Controller) C# Linux macOS Windows Cloud Service Web

Target Framework: .NET 5.0 (Current)

Authentication Type: None

- [x] Configure for HTTPS
- [ ] Enable Docker

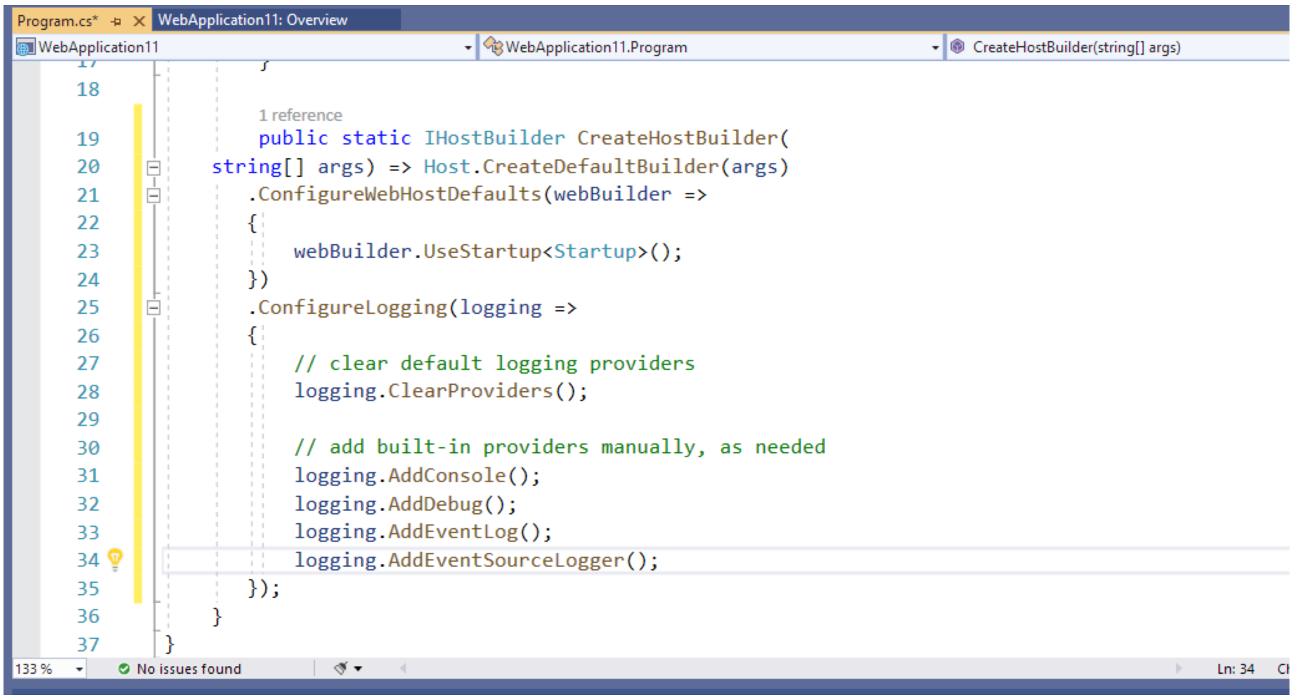
Docker OS: Linux

- [ ] Enable Razor runtime compilation

Back | Create

Step 3 –

To add your own set of logging providers, open the **Program.cs** file and can expand the call to CreateDefaultBuilder(), clear the default providers, and then add your own. The built-in providers now include **Console**, **Debug, EventLog** and **EventSource.**



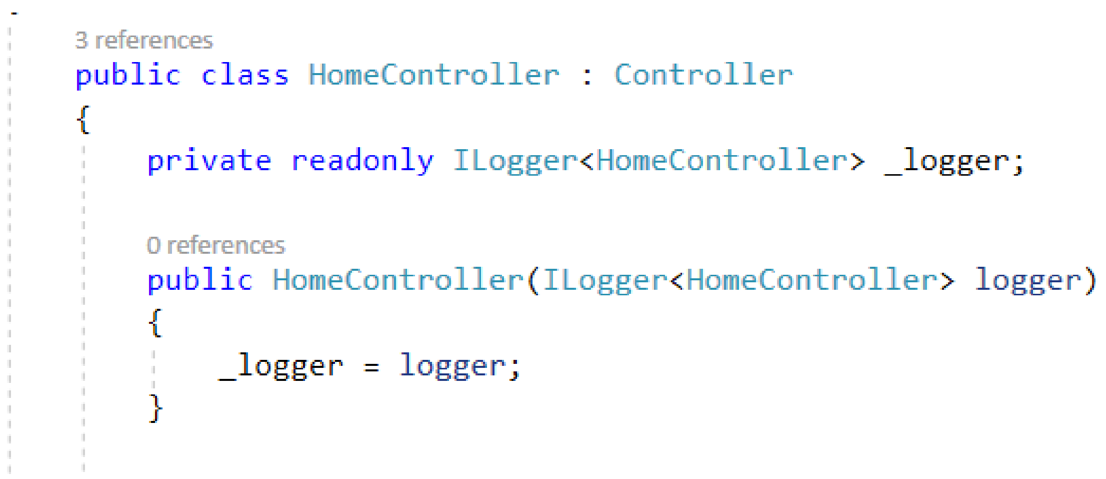
```
    public static IHostBuilder CreateHostBuilder(
string[] args) => Host.CreateDefaultBuilder(args)
    .ConfigureWebHostDefaults(webBuilder =>
    {
        webBuilder.UseStartup<Startup>();
    })
    .ConfigureLogging(logging =>
    {
        // clear default logging providers
        logging.ClearProviders();

        // add built-in providers manually, as needed
        logging.AddConsole();
        logging.AddDebug();
        logging.AddEventLog();
        logging.AddEventSourceLogger();
    });
}
}
```

Step 3 –

Open HomeController.cs or create your controller.

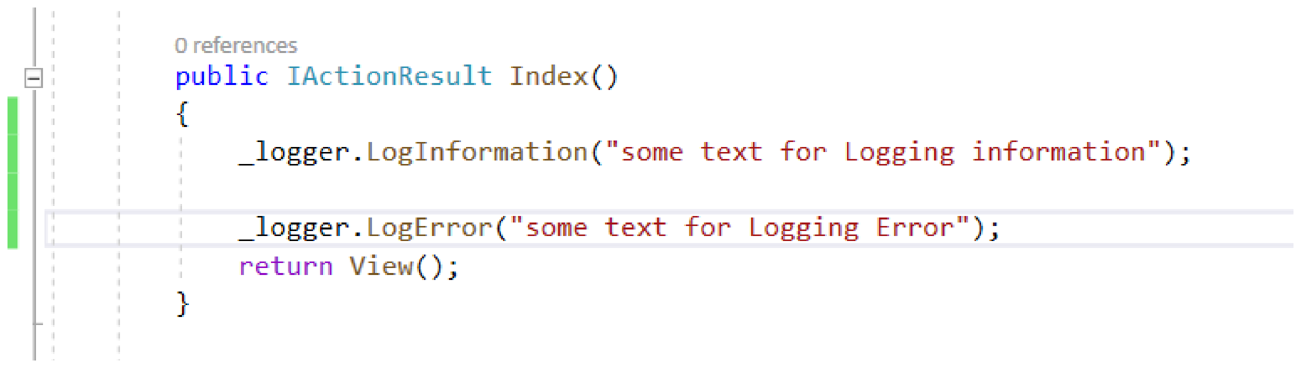
If you are creating your controller call the logger in the constructor like below



```
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;

    public HomeController(ILogger<HomeController> logger)
    {
        _logger = logger;
    }
```

Add the below code to the index function to log the data in the output window –

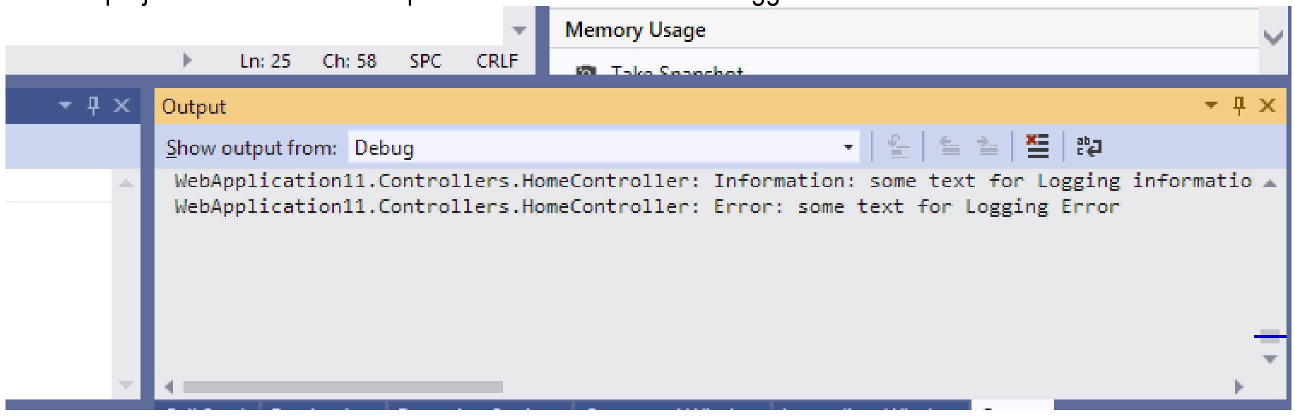


```
0 references
public IActionResult Index()
{
    _logger.LogInformation("some text for Logging information");

    _logger.LogError("some text for Logging Error");
    return View();
}
```

Step 4 –

Run the project and look at the output window to see the texts logged.

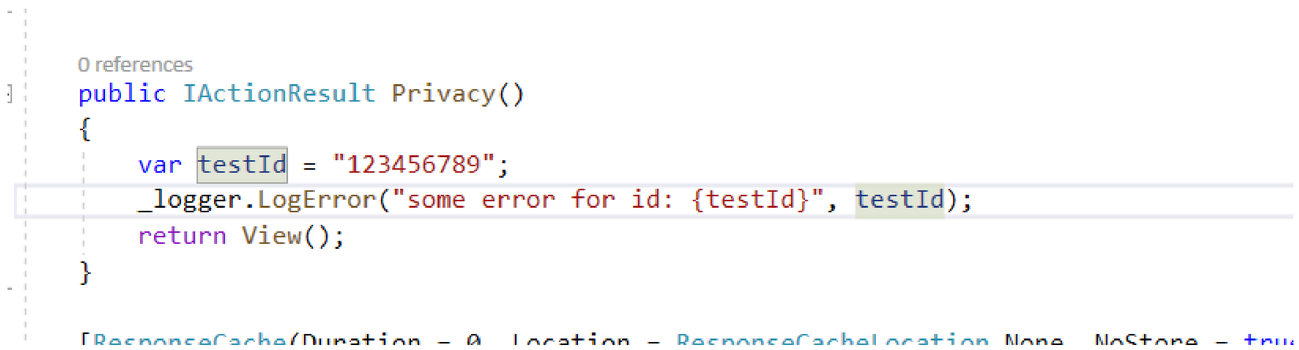


```
WebApplication11.Controllers.HomeController: Information: some text for Logging informatio
WebApplication11.Controllers.HomeController: Error: some text for Logging Error
```

You can add your logger and the logger type wherever necessary like in try-catch statement, exceptions etc.

Step 5 –

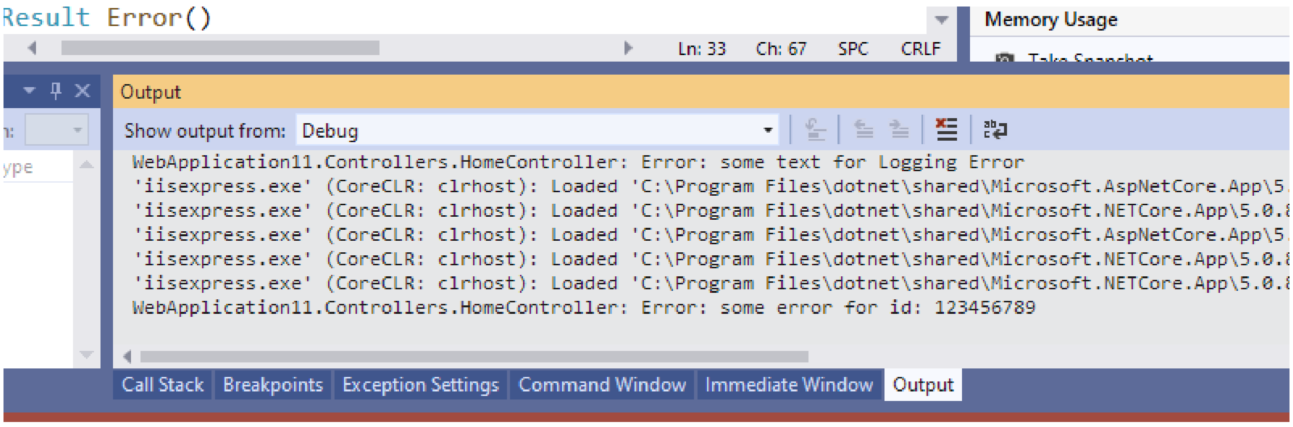
You can also query your log store for specific entries by searching for those arguments. Add the below code in the Privacy function.



```
0 references
public IActionResult Privacy()
{
    var testId = "123456789";
    _logger.LogError("some error for id: {testId}", testId);
    return View();
}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = tru
```

Run the project and click on the privacy tab on the website to view the log in the output console.



```
WebApplication11.Controllers.HomeController: Error: some text for Logging Error
'iisexpress.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App\5.
'iisexpress.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.
'iisexpress.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App\5.
'iisexpress.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.
'iisexpress.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\5.0.
WebApplication11.Controllers.HomeController: Error: some error for id: 123456789
```

THIRD-PARTY LOGGING – NLOG

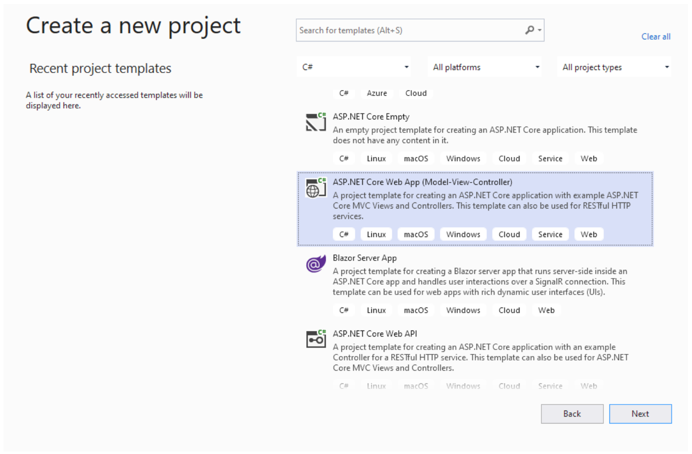
NLog is one of the top logging frameworks for dotnet. NLog is a .NET logging library that is both dependable and robust. NLog makes life easier for developers by providing a variety of write targets such as databases, files, consoles, and email. This readily qualifies it for inclusion in the list of the "Best Logging Frameworks for.NET in general. NLog comes with a variety of options that take logging to the next level. In addition, NLog's is flexible when it comes to changing the way the messages are logged. NLog is an Open Source Project and so is free to use.

The log levels are similar –

- Trace – The entire trace of the codebase.
- Debug – For debugging
- Info – A general Message
- Warn – Used for unexpected events and warnings.
- Error – When something breaks or exceptions.
- Fatal – When something very crucial breaks

Step 1 -

Create a new web application based on .Net Core 5



Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

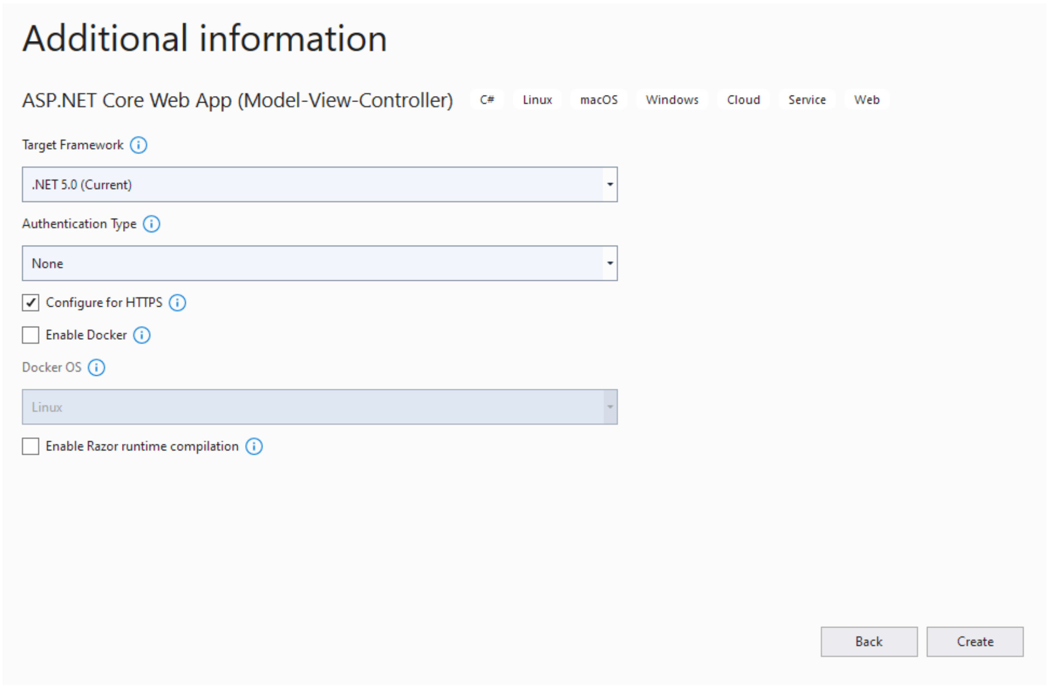
ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Blazor Server App
A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).

ASP.NET Core Web API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Search for templates (Alt+S)
Clear all
C#
All platforms
All project types
C# Azure Cloud
C# Linux macOS Windows Cloud Service Web
C# Linux macOS Windows Cloud Service Web
C# Linux macOS Windows Cloud Web
C# Linux macOS Windows Cloud Service Web
Back
Next

Provide the application name and select the target framework to be .NET 5.0 and click on create



Additional information

ASP.NET Core Web App (Model-View-Controller) C# Linux macOS Windows Cloud Service Web

Target Framework: .NET 5.0 (Current)

Authentication Type: None

[✓] Configure for HTTPS

[ ] Enable Docker

Docker OS: Linux

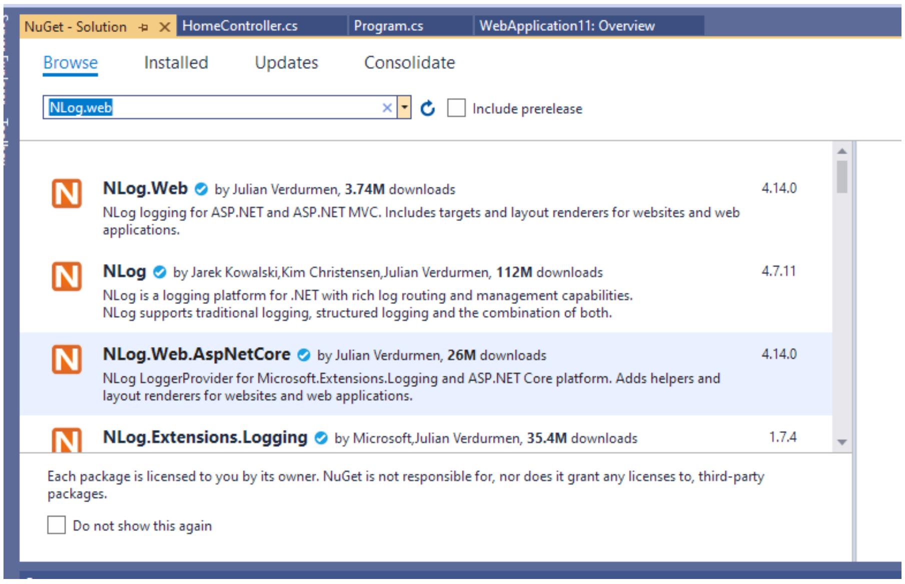
[ ] Enable Razor runtime compilation

Step 3 –

We need to install 2 Nuget packages. For that, go to Tools > NuGet Package Manager > Manage NuGet Packages for solution

Search for NLog.web in the browse tab and install the below 2 packages –

1) NLog
2) NLog.Web.AspNetCore



NLog.Web by Julian Verdurmen, 3.74M downloads 4.14.0
NLog logging for ASP.NET and ASP.NET MVC. Includes targets and layout renderers for websites and web applications.

NLog by Jarek Kowalski,Kim Christensen,Julian Verdurmen, 112M downloads 4.7.11
NLog is a logging platform for .NET with rich log routing and management capabilities. NLog supports traditional logging, structured logging and the combination of both.

NLog.Web.AspNetCore by Julian Verdurmen, 26M downloads 4.14.0
NLog LoggerProvider for Microsoft.Extensions.Logging and ASP.NET Core platform. Adds helpers and layout renderers for websites and web applications.

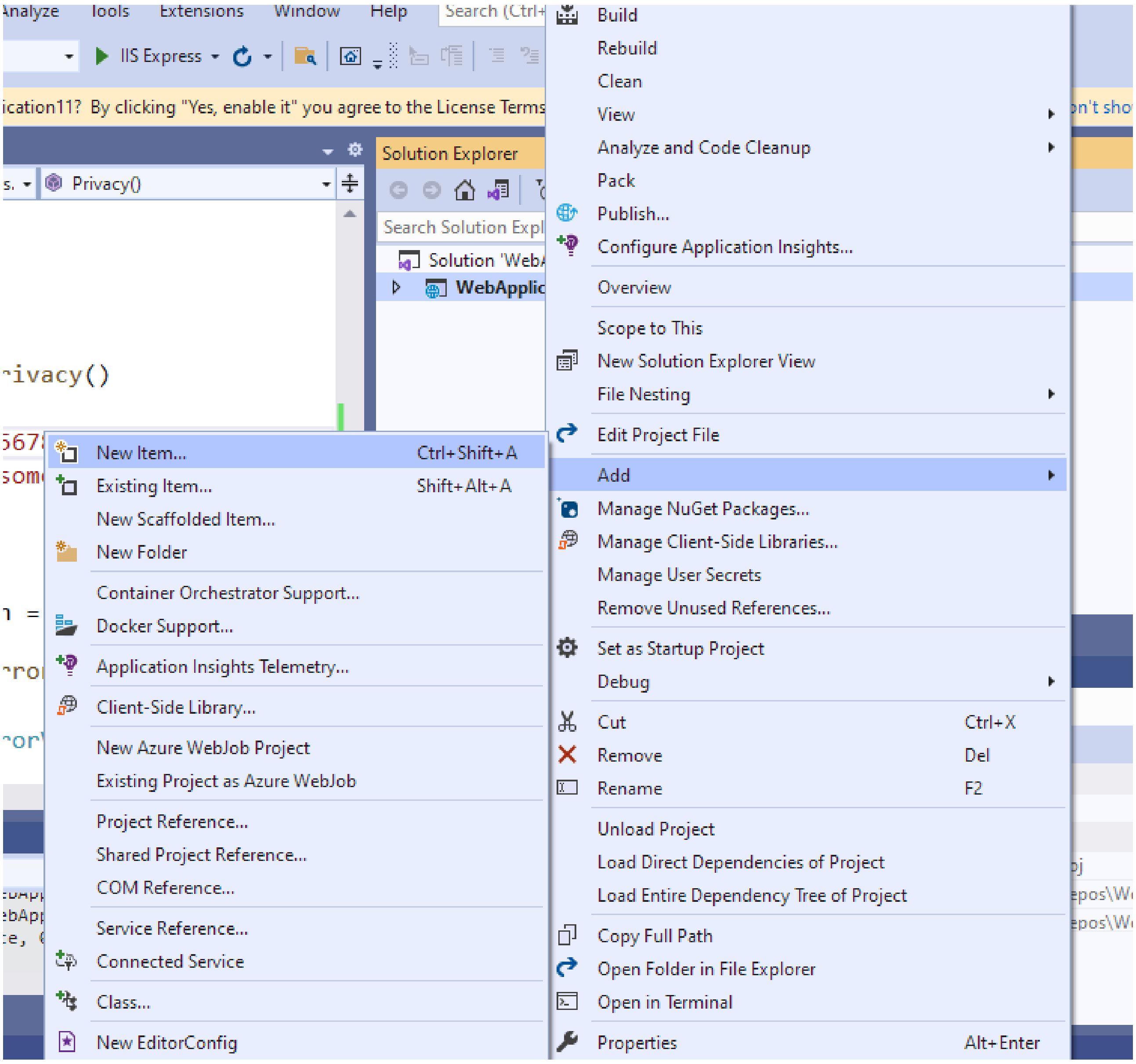
NLog.Extensions.Logging by Microsoft,Julian Verdurmen, 35.4M downloads 1.7.4

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

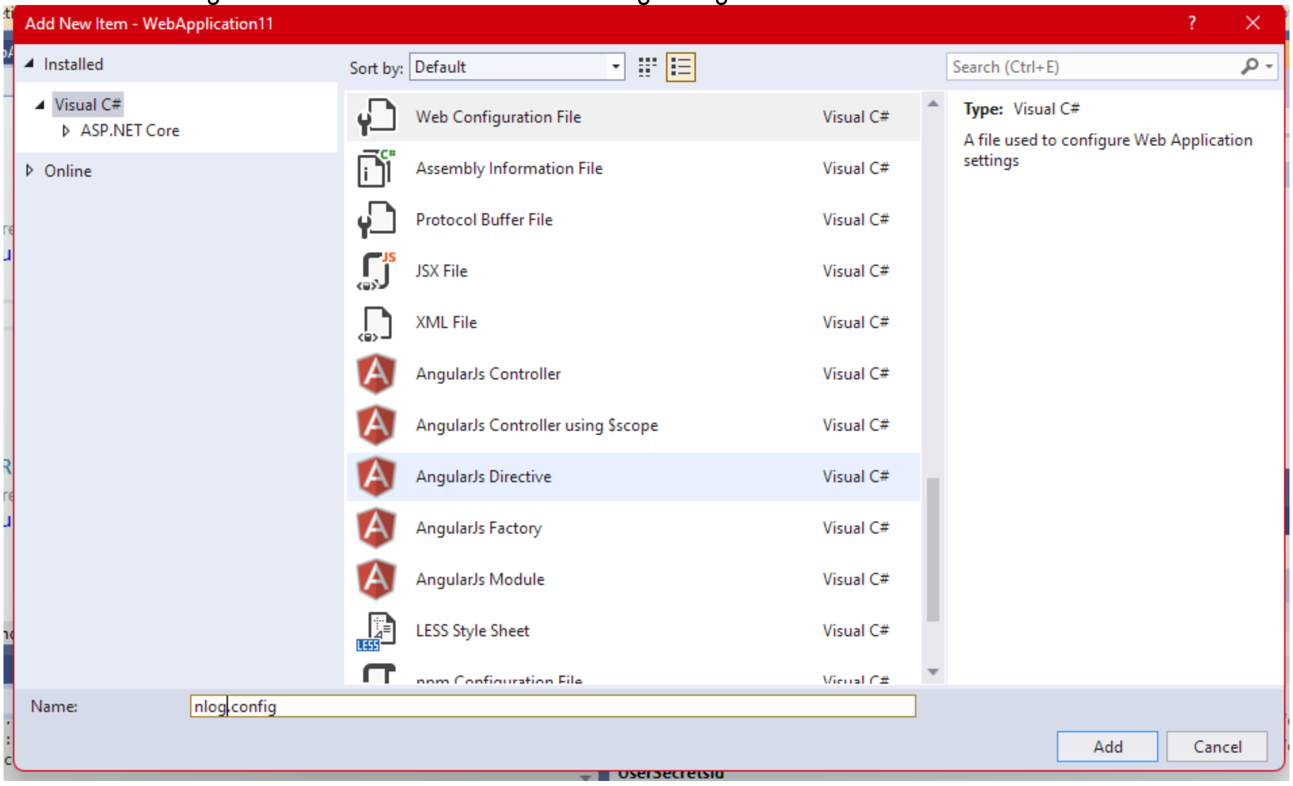
[ ] Do not show this again

Step 4 -

Create a file called nlog.config in the root folder. Right-click on the application name and click on Add


Analyze
Tools
Extensions
Window
Help
Search (Ctrl+
IIS Express
ication11? By clicking "Yes, enable it" you agree to the License Terms
Solution Explorer
Privacy()
Search Solution Expl
Solution 'Web
WebApplic
rivacy()
Build
Rebuild
Clean
View
Analyze and Code Cleanup
Pack
Publish...
Configure Application Insights...
Overview
Scope to This
New Solution Explorer View
File Nesting
Edit Project File
Add
Manage NuGet Packages...
Manage Client-Side Libraries...
Manage User Secrets
Remove Unused References...
Set as Startup Project
Debug
Cut Ctrl+X
Remove Del
Rename F2
Unload Project
Load Direct Dependencies of Project
Load Entire Dependency Tree of Project
Copy Full Path
Open Folder in File Explorer
Open in Terminal
Properties Alt+Enter
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Scaffolded Item...
New Folder
Container Orchestrator Support...
Docker Support...
Application Insights Telemetry...
Client-Side Library...
New Azure WebJob Project
Existing Project as Azure WebJob
Project Reference...
Shared Project Reference...
COM Reference...
Service Reference...
Connected Service
Class...
New EditorConfig

Select Web Configuration File and name the file as nlog.config -


Add New Item - WebApplication11
Installed
Visual C#
ASP.NET Core
Online
Sort by: Default
Search (Ctrl+E)
Web Configuration File Visual C#
Assembly Information File Visual C#
Protocol Buffer File Visual C#
JSX File Visual C#
XML File Visual C#
AngularJs Controller Visual C#
AngularJs Controller using $scope Visual C#
AngularJs Directive Visual C#
AngularJs Factory Visual C#
AngularJs Module Visual C#
LESS Style Sheet Visual C#
Add
Cancel

Type: Visual C#

A file used to configure Web Application settings

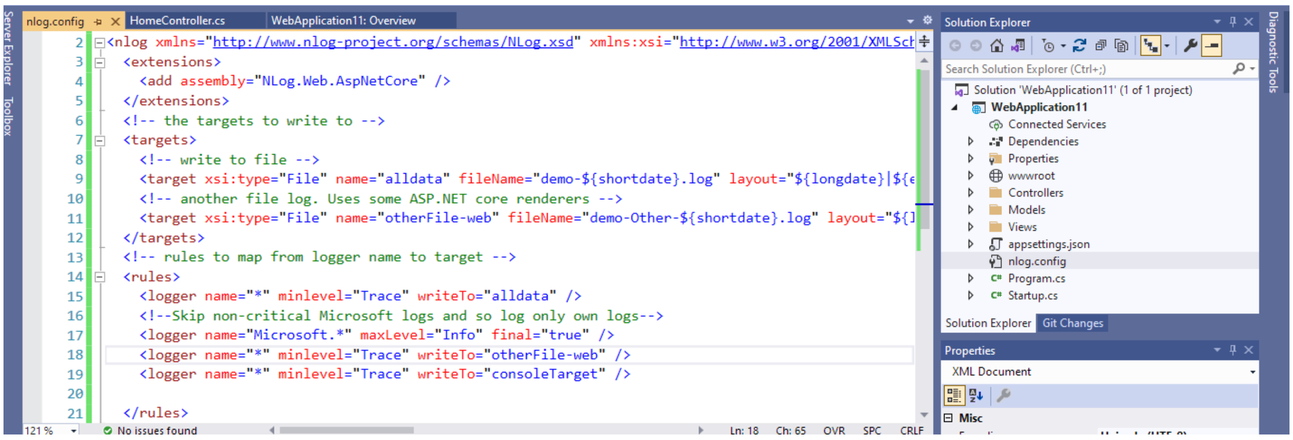
Name: nlog.config

Step 5 –

Copy the following code in the nlog.config file and save it

```
<?xml version="1.0" encoding="utf-8" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" autoReload="true" internalLogLevel="info"
internalLogFile="internalLog.txt">
  <extensions>
    <add assembly="NLog.Web.AspNetCore" />
  </extensions>
  <!-- the targets to write to -->
  <targets>
    <!-- write to file -->
    <target xsi:type="File" name="alldata" fileName="demo-${shortdate}.log" layout="${longdate}|${event-
properties:item=EventId_Id}|${uppercase:${level}}|${logger}|${message} ${exception:format=tostring}" />
    <!-- another file log. Uses some ASP.NET core renderers -->
    <target xsi:type="File" name="otherFile-web" fileName="demo-Other-${shortdate}.log"
layout="${longdate}|${event-properties:item=EventId_Id}|${uppercase:${level}}|${logger}|${message}
${exception:format=tostring}|url: ${aspnet-request-url}|action: ${aspnet-mvc-action}" />
```

```
    </targets>
    <!-- rules to map from logger name to target -->
    <rules>
        <logger name="*" minlevel="Trace" writeTo="alldata" />
        <!--Skip non-critical Microsoft logs and so log only own logs-->
        <logger name="Microsoft.*" maxLevel="Info" final="true" />
        <logger name="*" minlevel="Trace" writeTo="otherFile-web" />
    </rules>
</nlog>
```



```
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSch
  <extensions>
    <add assembly="NLog.Web.AspNetCore" />
  </extensions>
  <!-- the targets to write to -->
  <targets>
    <!-- write to file -->
    <target xsi:type="File" name="alldata" fileName="demo-${shortdate}.log" layout="${longdate}|${e
    <!-- another file log. Uses some ASP.NET core renderers -->
    <target xsi:type="File" name="otherFile-web" fileName="demo-Other-${shortdate}.log" layout="${]
  </targets>
  <!-- rules to map from logger name to target -->
  <rules>
    <logger name="*" minlevel="Trace" writeTo="alldata" />
    <!--Skip non-critical Microsoft logs and so log only own logs-->
    <logger name="Microsoft.*" maxLevel="Info" final="true" />
    <logger name="*" minlevel="Trace" writeTo="otherFile-web" />
    <logger name="*" minlevel="Trace" writeTo="consoleTarget" />

  </rules>
```

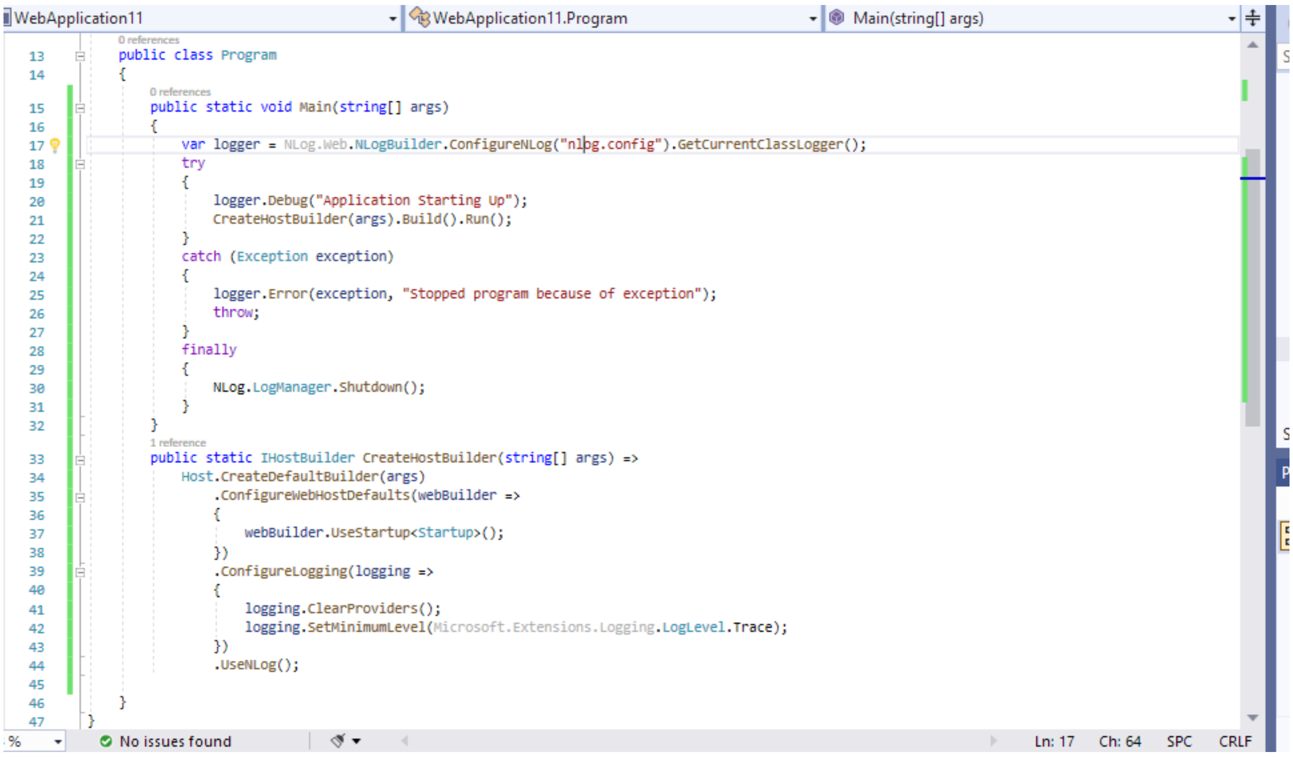
Your logs are written in targets. It could be as basic as a text file, or it could be an email that alerts the team when a critical problem happens. NLog gives you the ability to write to the following targets. NLog may be set up to write to many targets at once.

- Files
- Console & Colored Console
- Event Log
- Database
- Email

Step 6 –

Open Program.cs and make the following changes inside the program class. Also, add 'using NLog.Web;' at the top –

```
public static void Main(string[] args)
    {
        var logger = NLog.Web.NLogBuilder.ConfigureNLog("nlog.config").GetCurrentClassLogger();
        try
        {
            logger.Debug("Application Starting Up");
            CreateHostBuilder(args).Build().Run();
        }
        catch (Exception exception)
        {
            logger.Error(exception, "Stopped program because of exception");
            throw;
        }
        finally
        {
            NLog.LogManager.Shutdown();
        }
    }
    public static IHostBuilder CreateHostBuilder(string[] args) =>
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
                webBuilder.UseStartup<Startup>();
            })
            .ConfigureLogging(logging =>
            {
                logging.ClearProviders();
                logging.SetMinimumLevel(Microsoft.Extensions.Logging.LogLevel.Trace);
            })
            .UseNLog();
```

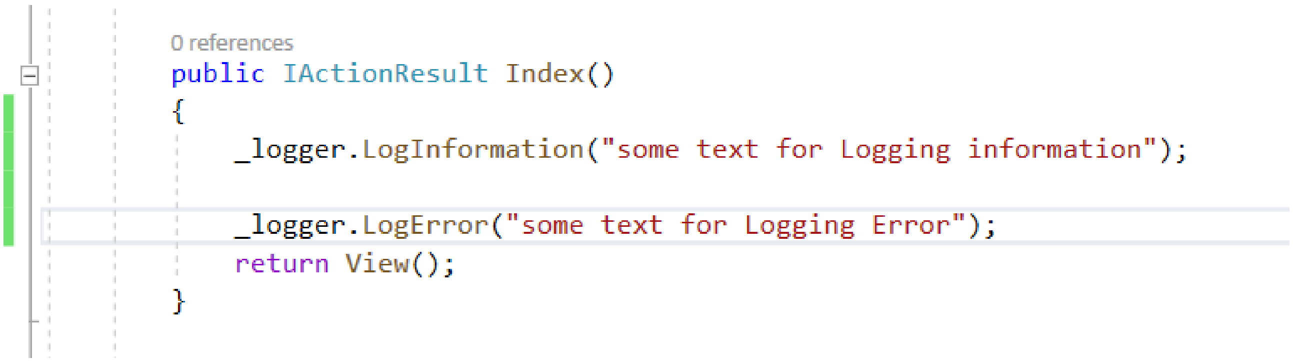


```
    0 references
    public class Program
    {
        0 references
        public static void Main(string[] args)
        {
            var logger = NLog.Web.NLogBuilder.ConfigureNLog("nlog.config").GetCurrentClassLogger();
            try
            {
                logger.Debug("Application Starting Up");
                CreateHostBuilder(args).Build().Run();
            }
            catch (Exception exception)
            {
                logger.Error(exception, "Stopped program because of exception");
                throw;
            }
            finally
            {
                NLog.LogManager.Shutdown();
            }
        }
        1 reference
        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                })
                .ConfigureLogging(logging =>
                {
                    logging.ClearProviders();
                    logging.SetMinimumLevel(Microsoft.Extensions.Logging.LogLevel.Trace);
                })
                .UseNLog();

    }
}
```

Step 7 –

Open HomeController.cs and modify the index function as below -



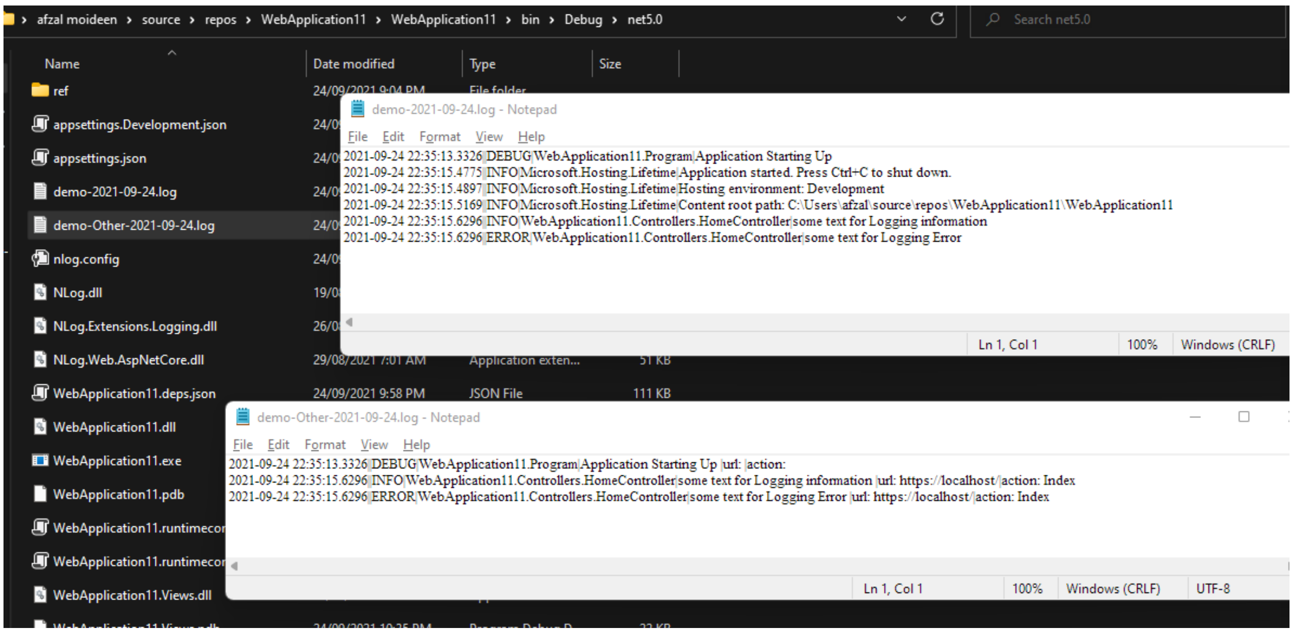
```
0 references
public IActionResult Index()
{
    _logger.LogInformation("some text for Logging information");

    _logger.LogError("some text for Logging Error");
    return View();
}
```

This information will now be saved into a log file inside the debug folder. Open your file explorer and go to bin\Debug\net5.0 to find the file with the name that we gave in the configuration file nlog.config.
In this activity, we are writing it to 2 files specified with filenames as

fileName="demo-${shortdate}.log" and fileName="demo-Other-${shortdate}.log"

Opening these files will show that the logs have been written based on your configuration on what to write in them


afzal moideen › source › repos › WebApplication11 › WebApplication11 › bin › Debug › net5.0
Search net5.0
Name
Date modified
Type
Size
ref
appsettings.Development.json
appsettings.json
demo-2021-09-24.log
demo-Other-2021-09-24.log
nlog.config
NLog.dll
NLog.Extensions.Logging.dll
NLog.Web.AspNetCore.dll
WebApplication11.deps.json
WebApplication11.dll
WebApplication11.exe
WebApplication11.pdb
WebApplication11.runtimecor
WebApplication11.runtimecor
WebApplication11.Views.dll
demo-2021-09-24.log - Notepad
File Edit Format View Help
Ln 1, Col 1
100%
Windows (CRLF)
demo-Other-2021-09-24.log - Notepad
UTF-8

```
2021-09-24 22:35:13.3326|DEBUG|WebApplication11.Program|Application Starting Up
2021-09-24 22:35:15.4775|INFO|Microsoft.Hosting.Lifetime|Application started. Press Ctrl+C to shut down.
2021-09-24 22:35:15.4897|INFO|Microsoft.Hosting.Lifetime|Hosting environment: Development
2021-09-24 22:35:15.5169|INFO|Microsoft.Hosting.Lifetime|Content root path: C:\Users\afzal\source\repos\WebApplication11\WebApplication11
2021-09-24 22:35:15.6296|INFO|WebApplication11.Controllers.HomeController|some text for Logging information
2021-09-24 22:35:15.6296|ERROR|WebApplication11.Controllers.HomeController|some text for Logging Error
```

```
2021-09-24 22:35:13.3326|DEBUG|WebApplication11.Program|Application Starting Up |url: |action:
2021-09-24 22:35:15.6296|INFO|WebApplication11.Controllers.HomeController|some text for Logging information |url: https://localhost/|action: Index
2021-09-24 22:35:15.6296|ERROR|WebApplication11.Controllers.HomeController|some text for Logging Error |url: https://localhost/|action: Index
```

Using NLog, one can set what target to write to and what content is to be logged.

References

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

https://nlog-project.org/