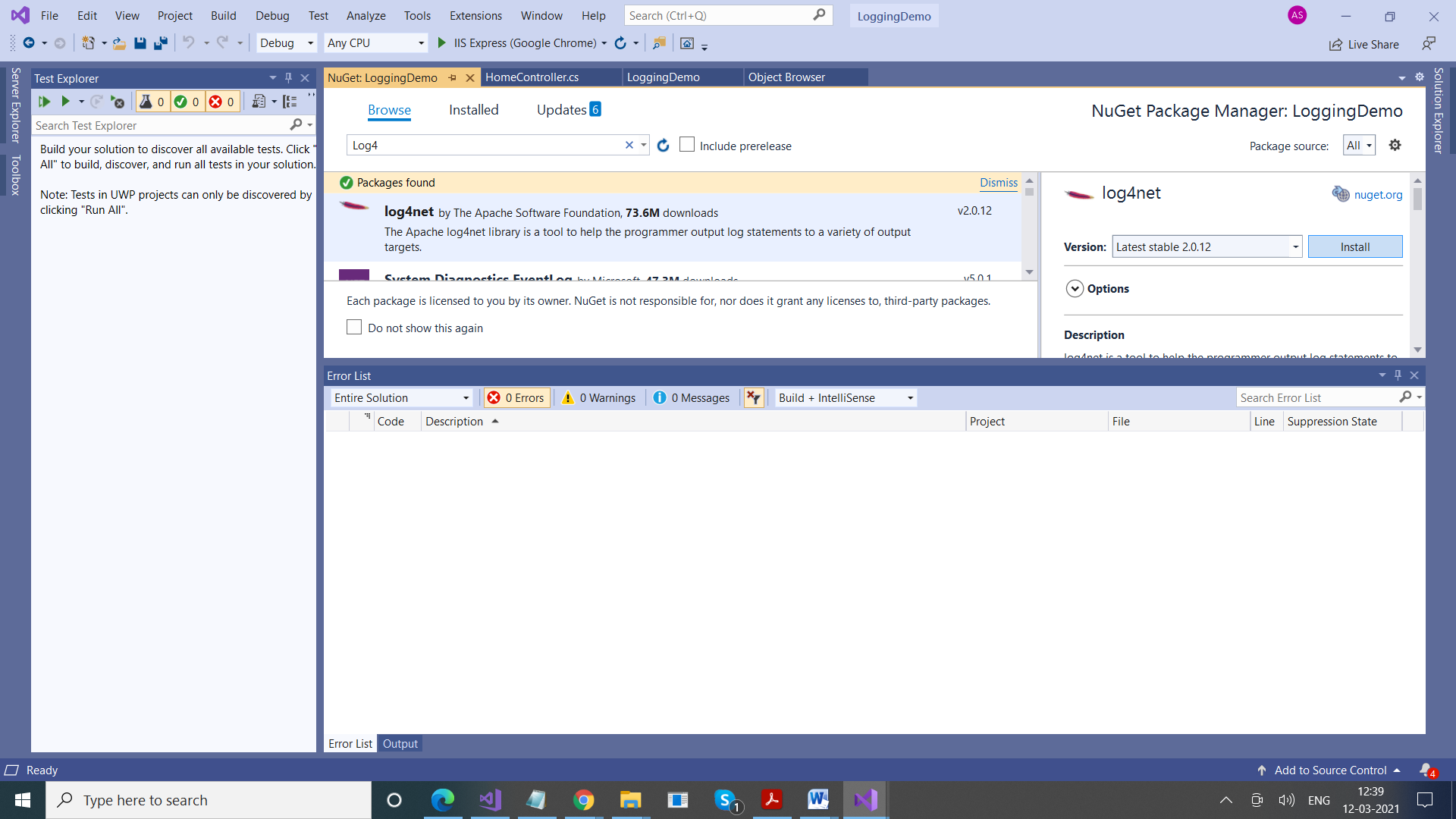
Step 1: Install log4net package



Step 2:

Add log4net.Config.XmlConfigurator.Configure(); in global.asax file

protected void Application\_Start()

{

AreaRegistration.RegisterAllAreas();

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

**log4net.Config.XmlConfigurator.Configure();**

}

Step 3: Add an entry for log4net in web.config file

<configSections>

<section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler, log4net" />

</configSections>

<log4net debug="false">

<appender name="LogFileAppender" type="log4net.Appender.FileAppender">

<param name="File" value="E:\test\test.log" />

<!--<param name="AppendToFile" value="true"/>-->

<layout type="log4net.Layout.PatternLayout">

<param name="ConversionPattern" value="%d [%t] %-5p %c %m%n" />

</layout>

</appender>

<root>

<level value="All" />

<appender-ref ref="LogFileAppender" />

</root>

</log4net>

Step 4:

We want to log messages from Index Action Method of HomeController

public class HomeController : Controller

{

log4net.ILog logger = log4net.LogManager.GetLogger(typeof(HomeController)); //Declaring Log4Net

public ActionResult Index()

{

try

{

int x, y, z;

x = 5; y = 0;

z = x / y;

}

catch (Exception ex)

{

logger.Error(ex.ToString());

}

return View();

}

Step 5: Check log file in your system

In case , you want to log for entire application, follow these steps

Create a folder Utilities, in that add a Class Log.cs

using log4net;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace LoggingDemo.Utilities

{

public class Log

{

private static readonly Log \_instance = new Log();

protected ILog monitoringLogger;

protected static ILog debugLogger;

private Log()

{

monitoringLogger = LogManager.GetLogger("MonitoringLogger");

debugLogger = LogManager.GetLogger("DebugLogger");

}

public static void Debug(string message)

{

debugLogger.Debug(message);

}

/// <summary>

///

/// </summary>

/// <param name="message">The object message to log</param>

/// <param name="exception">The exception to log, including its stack trace </param>

public static void Debug(string message, System.Exception exception)

{

debugLogger.Debug(message, exception);

}

/// <summary>

///

/// </summary>

/// <param name="message">The object message to log</param>

public static void Info(string message)

{

\_instance.monitoringLogger.Info(message);

}

/// <summary>

///

/// </summary>

/// <param name="message">The object message to log</param>

/// <param name="exception">The exception to log, including its stack trace </param>

public static void Info(string message, System.Exception exception)

{

\_instance.monitoringLogger.Info(message, exception);

}

/// <summary>

///

/// </summary>

/// <param name="message">The object message to log</param>

public static void Warn(string message)

{

\_instance.monitoringLogger.Warn(message);

}

/// <summary>

///

/// </summary>

/// <param name="message">The object message to log</param>

/// <param name="exception">The exception to log, including its stack trace </param>

public static void Warn(string message, System.Exception exception)

{

\_instance.monitoringLogger.Warn(message, exception);

}

/// <summary>

///

/// </summary>

/// <param name="message">The object message to log</param>

public static void Error(string message)

{

\_instance.monitoringLogger.Error(message);

}

/// <summary>

///

/// </summary>

/// <param name="message">The object message to log</param>

/// <param name="exception">The exception to log, including its stack trace </param>

public static void Error(string message, System.Exception exception)

{

\_instance.monitoringLogger.Error(message, exception);

}

/// <summary>

///

/// </summary>

/// <param name="message">The object message to log</param>

public static void Fatal(string message)

{

\_instance.monitoringLogger.Fatal(message);

}

/// <summary>

///

/// </summary>

/// <param name="message">The object message to log</param>

/// <param name="exception">The exception to log, including its stack trace </param>

public static void Fatal(string message, System.Exception exception)

{

\_instance.monitoringLogger.Fatal(message, exception);

}

}

}

Now, you need to call the logger class that logs directly into your action method.

public ActionResult Index()

{

Log.Info("Login-page started...");

return View();

}

**LOG4NET**

Before showing what Log4Net is I would like to explaing what logging actually is.

**Logging**

In computing, a **logfile** is a **file** that records either events that occur in an operating system or in a programme or when a software runs.

**Logging** is the act of keeping a **log.**

Or we can say, logging is a method of tracking/monitoring what is going on when an application is running. Log records will be the most needed items when something goes wrong in your application.

Now I am explaing Log4Net.

**LOG4NET**

Log4net is an open-source library that allows or helps the .NET applications to trace the details of the errors that have occurred in a project.

Log4Net is a framework for implementing logging mechanisms. It is an open source framework.

Log4net provides a simple mechanism for logging information to a variety of sources. Information is logged via one or more loggers. These loggers are provided at the below levels of logging:

* Debug
* Information
* Warnings
* Error
* Fatal

**APPENDER: The <appender /> element specifies the name and logger type. It specifies where the information will be logged, how it will be logged and under what circumstances the information will be logged. You can check the various types of appender in the link**[**http://logging.apache.org/log4net/release/config-examples.html.**](http://logging.apache.org/log4net/release/config-examples.html)

**param name:** specifies the file name and path where the log will be saved. In this case it is "test.log".

**layout:**The Layout element tells Log4Net how each log line should look like.

**Root: You need to have one root section to house your top-level logger references. These are the loggers that inherit information from your base logger (root).**

<log4net>

<!-- appenders go here -->

<root>

<level value="INFO" />

<appender-ref ref="myLogAppender" />

</root>

<logger name="MyNamespace.Foo.Bar">

<level value="DEBUG" />

</logger>

</log4net>

**Different Appender**

* Rolling File Appender - It writes to the output window or the command window.
* File Appender -This appender will write to a text file.
* ADO.NET Appender -This appender will write to a Database.
* Console Appender-This appender performs the same functions as the file appender but with the additional option to store a certain amount of data only before starting a new log file.