logging using log4net

TABLE OF CONTENTS

Introduction 3

Why is Logging important? 3

Log4Net 3

Implementation 3

Step 1: Installation 3

Step 2: Create a new file log4net.config 5

Step 3: Configure logging in program.cs file 5

Step 4: Logging 5

Configuration 6

Logger 6

Appenders 7

Layout 8

Conclusion 9

Janani Priya T

# Introduction

This documentation provides guidance for implementing logging in .NET framework using Log4Net, procedure for installation, configuration, logging, and information about logger, appender and layout.

## Why is logging important?

Logging is important in any application as it makes debugging easier and facilitates monitoring in production environment. In development environment it provides context about the project execution which can be analyzed to accelerate the process of debugging. In production environment, effective logging aids in monitoring an application, diagnosing, and fixing issues. It becomes a source of execution details which can be referred to diagnose in case of a crash.

Logging in .NET framework can be done with the help of Log4Net.

## LOG4NET

The Apache log4net library is a tool to help the programmer output log statements to a variety of output targets. It supports multiple framework such as .NET and Mono. It provides features such as

* Support for multiple frameworks
* Output to multiple logging targets
* Hierarchical logging architecture
* XML Configuration
* Dynamic Configuration
* Logging Context
* Proven architecture
* Modular and extensible design
* High performance with flexibility

# implementation

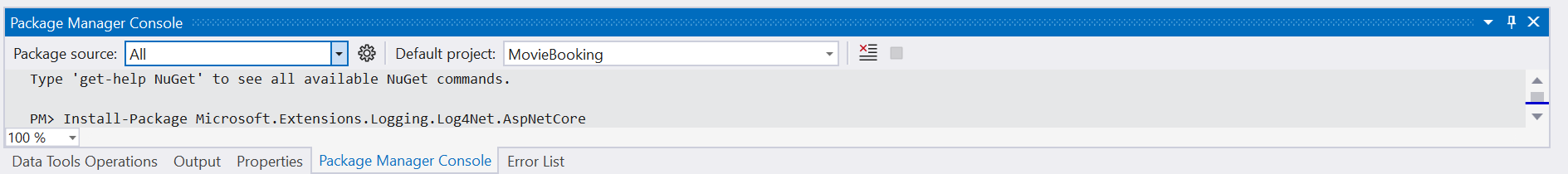
## Step 1: Installation

To install log4net, in visual studio go to Tools -> NuGet Package Manager -> Package Manager Console, and then run the following command

Install-Package Microsoft.Extensions.Logging.Log4Net.AspNetCore

Graphical user interface, application

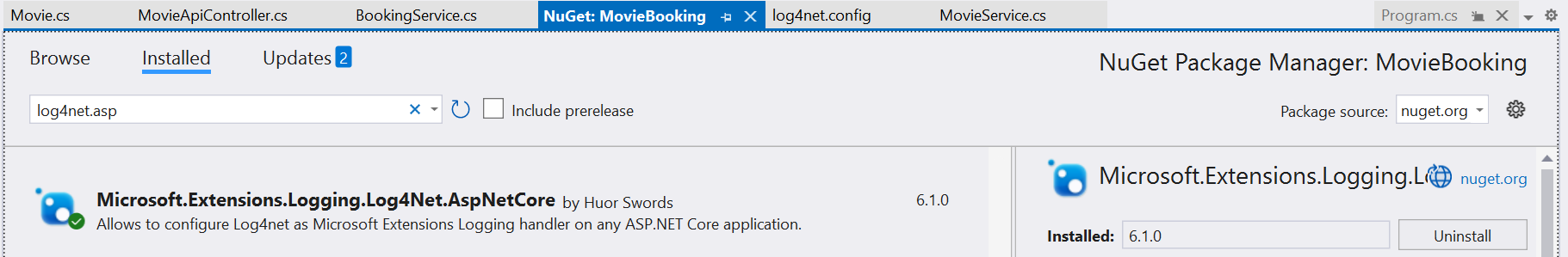
Description automatically generated



Or In solution Explorer, right click on the project and select Manage NuGet Packages and browse for Microsoft.Extensions.Logging.Log4Net.AspNetCore and install it.

Graphical user interface, application

Description automatically generated



This installs Log4Net logging extension for ASP.NET Core and the Log4Net library.

Step 2: Create a new file log4net.config

In your project create a new file log4net.config and add the configuration details as shown below. Configuration options are discussed in detail in the following sections.

Graphical user interface, text, application

Description automatically generated

Step 3: Configure logging in program.cs file

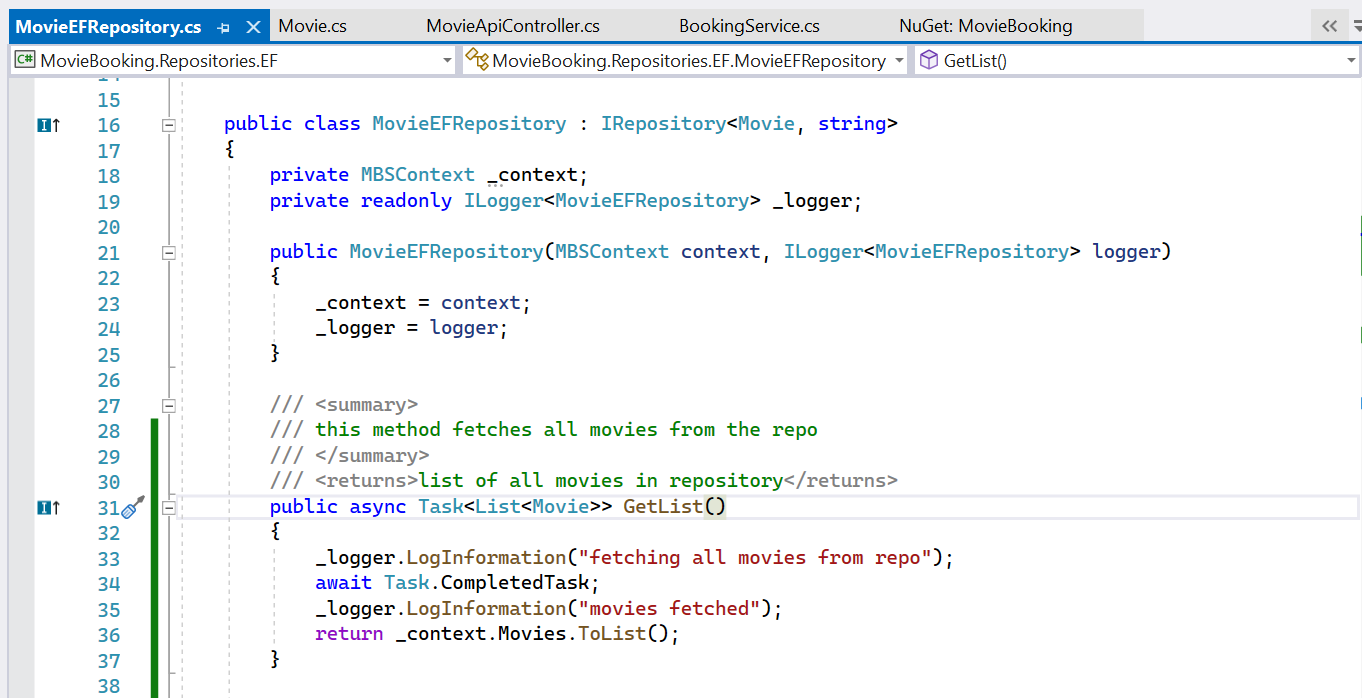
In program.cs file add the below given code to configure logging using log4net with log4net.config file

Graphical user interface, text, application

Description automatically generated

Step 4: Logging

To perform logging create an ILogger instance. Log4Net provides logging methods for each log level (discussed in next section). Implement logging as per requirements such us entry to a method or successful completion of an action, etc.



Log messages are added to the log file created.

Scatter chart

Description automatically generated

# Configuration

Log4net has three main components: loggers, appenders and layouts. These three types of components work together to enable developers to log messages according to message type and level, and to control at runtime how these messages are formatted and where they are reported.

## Logger

**Logger hierarchy**

One of the features of log4net is its ability to disable certain log statements while allowing others to print unhindered. This can be achieved by named hierarchy. The root logger resides at the top of the logger hierarchy, and we can create our own loggers and define their hierarchy.

A logger is said to be an ancestor of another logger if its name followed by a dot is a prefix of the descendant logger name. A logger is said to be a parent of a child logger if there are no ancestors between itself and the descendant logger.

Loggers can be assigned with levels, which are instances of the log4net.Core.Level class. The levels defined in order of increasing priority are :

* All
* Debug
* Info
* Warn
* Error
* Fatal
* Off

**Level Inheritance**

If a logger is not assigned with a logger level, then it inherits from its closest ancestor. If a logger X is not assigned with a logger level then its logger level will be the first non-null level in the logger hierarchy proceeding upwards from x towards the root logger. To ensure every logger is assigned with a level, the root logger is always assigned with a level which has a default value debug.

## Appenders

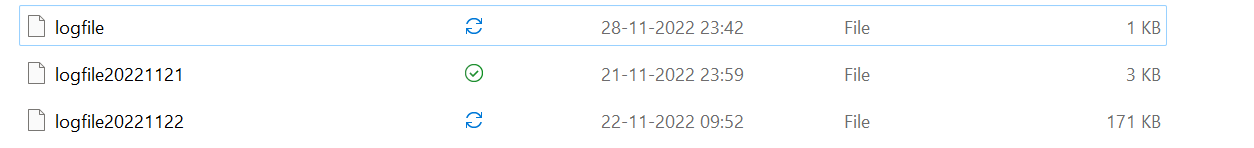
Log4net allows output logging requests to multiple destinations called appenders. Appenders must implement the log4net.Appenders.IAppender interface.

In the previous implementation we have used **log4net.Appender.RollingFileAppender.** It writes logging events to a file in the file system. The RollingFileAppender can be configured to log to multiple files based upon date or file size constraints.

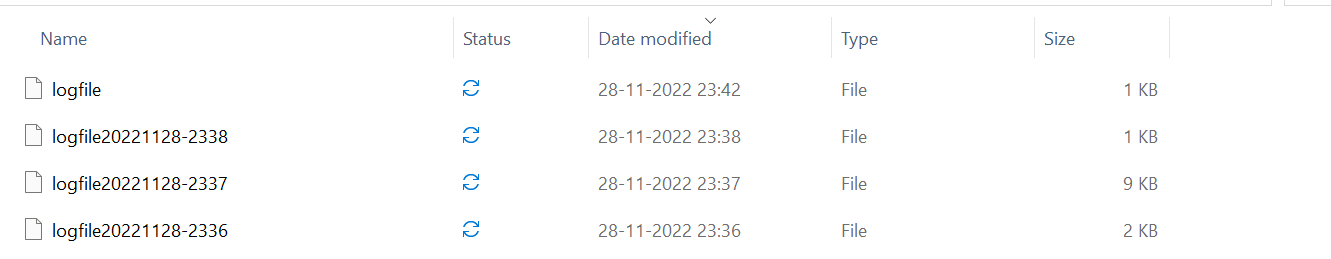
Graphical user interface, text, application

Description automatically generated

The rolling period can be changed by adjusting the date pattern value. Now the date pattern yyyyMMdd rolls every day.



If you change this value to yyyyMMdd-HHmm, then this will roll the log file every minute.

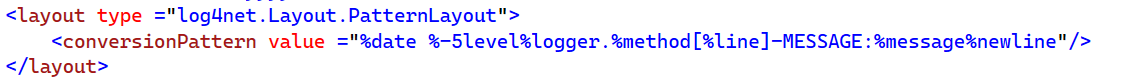


Log4Net provides multiple other appenders to log to different targets.

## Layouts

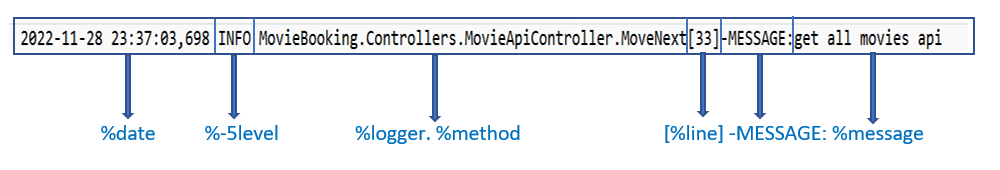
Layout specifies how the logging data will be written. The PatternLayout, part of the standard log4net distribution, lets the user specify the output format according to conversion patterns.

In the above implementation we have used log4net.Layout.PatternLayout.



**Conversion Patterns**

* %date - Outputs the date using the local time zone information. This date can be formatted using the curly braces and a layout pattern such as %date{MMMM dd, yyyy HH:mm:ss, fff} to output the value in the format "November 22, 2022 02:01:43, 767".
* %level – outputs the level of the logging event.
* -5 - Right pad with spaces if the level name is less than 5 characters long.
* %logger - outputs the logger of the logging event.
* %method - outputs the method name where the logging request was issued.
* %line - outputs the line number from where the logging request was issued.
* %message - outputs the application supplied message associated with the logging event.
* %newline - outputs the platform dependent line separator character or characters.



# Conclusion

Log4Net provides an easy implementation of logging in .NET framework. The documentation has provided guidance for implementing Log4Net and has discussed logger and its features such as hierarchical logger and log levels, appender, layout and conversion pattern.