# USING A SINGLE LOG4NET CONFIGURATION FILE ACROSS A MULTI-TIER APPLICATION

log4net is a tool to help the programmer output log statements to a variety of output targets.

This example is written in Visual Studio 2015 for a WebAPI2 solution. The sample solution consists of four projects that include a project for data, models, unit tests, etc.

SETUP:

1. Create your project and then import the following Nuget package: <https://www.nuget.org/packages/log4net/>
2. For each project, add reference to log4net.dll assembly. Though all configuration will be in the main API project, other projects will require the assembly reference as well.
3. Configuration files to be included are:
   1. Log4net.config
   2. Web.config
   3. Settings.xml
   4. Global.asax.cs
4. Note that the <appsettings> node of the web.config file has been abstracted into a separate file --- Settings.xml. These settings will be referenced in the web.config in the following line of code:

<appSettings file="c:\webapps\log4sample\config\settings.xml"/>

SECTION:

1. In your web.config file, enter the following:

<configSections>

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

</configSections>

1. Now, add the logging configuration values. These can be added in your web.config(or app.config) but in this scenario, it has its own configuration file – log4net.config:

<log4net>

<appender name="RollingFile" type="log4net.Appender.RollingFileAppender">…..

</log4net>

1. Then, you will want to add the path to the file in your <appSettings> node:

<add key="logPath" value="c:\webapps\log4sample\config\log4net.config" />

1. In your Global.asax file, add the following to the Application\_Start() method:

//log4net setup:

var logPath = (ConfigurationManager.AppSettings["logPath"]);

log4net.Config.XmlConfigurator.ConfigureAndWatch(new FileInfo(logPath));

REFERENCES:

<http://geekswithblogs.net/michaelstephenson/archive/2014/01/02/155044.aspx>

<https://logging.apache.org/log4net/release/config-examples.html>

EXTRA:

You can also add a helper method to customize your logging and add details that would help with your debugging:

public class Log4NetHelper

{

public string LogMsg(Message m, [Optional] string exceptionMsg, [CallerMemberName] string methodName = "")

{

string msg = $"{methodName} {m} {exceptionMsg}";

return msg;

}

}

To reference:

private readonly Log4NetHelper \_msg = new Log4NetHelper();

\_logger.Debug(\_msg.LogMsg(Log4NetHelper.Message.InitialRequestStart));



