**Tipo         :** Guía del Enunciado

**Capitulo  :** Creando Aplicaciones ASP.NET Core

**Duración :** 180 minutos

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

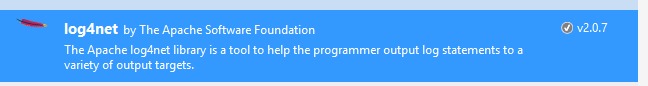
1. **OBJETIVO**

Crear aplicaciones Web MVC utilizando las plantillas que proporciona Visual Studio.

1. **REQUISITOS**

Los siguientes elementos de software son necesarios para la realización del  laboratorio:

1. Windows 10 (como mínimo Windows 8)
2. Visual Studio 2015 (como mínimo Visual Studio 2013)
3. **EJECUCIÓN DEL LABORATORIO**
   1. **Implementar Log4Net in AspNet Core**
4. En el proyecto “Cibertec.Web”, instalar el nuget de “log4Net”



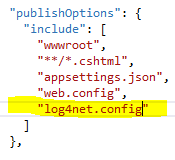
2. En el proyecto “Cibertec.Web”, crear un archivo xml llamado “log4net.xml” y copiar el siguiente código:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <log4net>   <appender name="RollingFile" type="log4net.Appender.FileAppender">     <file value="C:\Temp\app.log" />     <layout type="log4net.Layout.PatternLayout">       <conversionPattern value="%-5p %d{hh:mm:ss} %message%newline" />     </layout>   </appender>    <root>     <level value="ALL" />     <appender-ref ref="RollingFile" />   </root>  </log4net> |

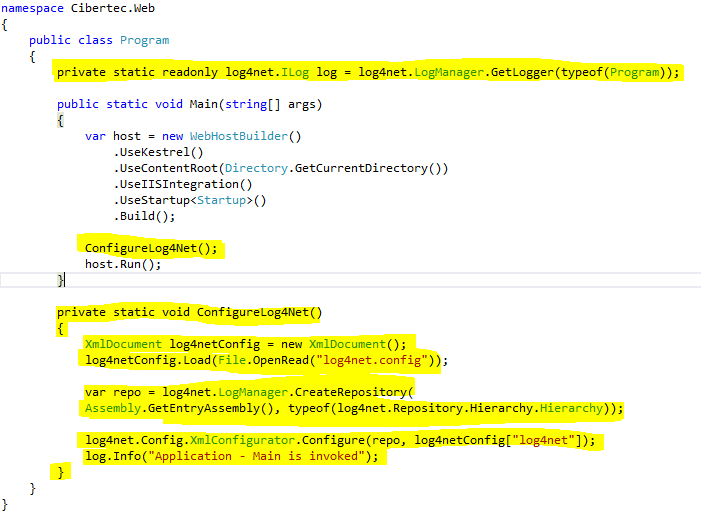
3. En el proyecto “Cibertec.Web”,  en el archivo “project.json”, en esta parte



agregar el nuevo archivo de tal manera que quede de la siguiente manera:



4. En el proyecto “Cibertec.Web”, en el archivo “Program.cs”, debe quedar de la siguiente manera, se debe agregar la parte resaltada:



5. Ahora podemos ejecutar la aplicacion, y veremos el archivo log generado en la ruta especificada.

6. Ahora necesitamos adaptar un provider y un metodo de extension para tener un mejor detalle de todo. Creamos una carpeta llamada log.

https://lh6.googleusercontent.com/My38j8DC2bfP0alK1-TkJUgITLdkkZAeQqETPa2RVXaXZ54wYtjpjCJcg7Nwd7-PFwNEaj-22m8Jeq6m1Sgq_pGXjWHySoA1AvV2N1NO9EELeCcCT2ymfUepUsS8QqAuGkq_jigV

7. Creamos la clase “Log4NetLogger”, debe quedar de la siguiente manera:

|  |
| --- |
| public class Log4NetLogger : ILogger     {         private readonly string \_name;         private readonly XmlElement \_xmlElement;         private readonly ILog \_log;         private ILoggerRepository \_loggerRepository;         public Log4NetLogger(string name, XmlElement xmlElement)         {             \_name = name;             \_xmlElement = xmlElement;             \_loggerRepository = log4net.LogManager.CreateRepository(                 Assembly.GetEntryAssembly(), typeof(log4net.Repository.Hierarchy.Hierarchy));             \_log = LogManager.GetLogger(\_loggerRepository.Name, name);             log4net.Config.XmlConfigurator.Configure(\_loggerRepository, xmlElement);         }         public IDisposable BeginScope<TState>(TState state)         {             return null;         }         public bool IsEnabled(LogLevel logLevel)         {             switch (logLevel)             {                 case LogLevel.Critical:                     return \_log.IsFatalEnabled;                 case LogLevel.Debug:                 case LogLevel.Trace:                     return \_log.IsDebugEnabled;                 case LogLevel.Error:                     return \_log.IsErrorEnabled;                 case LogLevel.Information:                     return \_log.IsInfoEnabled;                 case LogLevel.Warning:                     return \_log.IsWarnEnabled;                 default:                     throw new ArgumentOutOfRangeException(nameof(logLevel));             }         }         public void Log<TState>(LogLevel logLevel, EventId eventId, TState state,             Exception exception, Func<TState, Exception, string> formatter)         {             if (!IsEnabled(logLevel))             {                 return;             }             if (formatter == null)             {                 throw new ArgumentNullException(nameof(formatter));             }             string message = null;             if (null != formatter)             {                 message = formatter(state, exception);             }             if (!string.IsNullOrEmpty(message) || exception != null)             {                 switch (logLevel)                 {                     case LogLevel.Critical:                         \_log.Fatal(message);                         break;                     case LogLevel.Debug:                     case LogLevel.Trace:                         \_log.Debug(message);                         break;                     case LogLevel.Error:                         \_log.Error(message);                         break;                     case LogLevel.Information:                         \_log.Info(message);                         break;                     case LogLevel.Warning:                         \_log.Warn(message);                         break;                     default:                         \_log.Warn($"Encountered unknown log level {logLevel}, writing out as Info.");                         \_log.Info(message, exception);                         break;                 }             }         }     } |

8. Ahora agregamos la clase “Log4NetProvider”:

|  |
| --- |
| public class Log4NetProvider : ILoggerProvider     {         private readonly string \_log4NetConfigFile;         private readonly ConcurrentDictionary<string, Log4NetLogger> \_loggers =             new ConcurrentDictionary<string, Log4NetLogger>();         public Log4NetProvider(string log4NetConfigFile)         {             \_log4NetConfigFile = log4NetConfigFile;         }         public ILogger CreateLogger(string categoryName)         {             return \_loggers.GetOrAdd(categoryName, CreateLoggerImplementation);         }         public void Dispose()         {             \_loggers.Clear();         }         private Log4NetLogger CreateLoggerImplementation(string name)         {             return new Log4NetLogger(name, Parselog4NetConfigFile(\_log4NetConfigFile));         }         private static XmlElement Parselog4NetConfigFile(string filename)         {             XmlDocument log4netConfig = new XmlDocument();             log4netConfig.Load(File.OpenRead(filename));             return log4netConfig["log4net"];         }     } |

9. Agregamos la siguiente Clase “Log4netExtensions”

|  |
| --- |
| public static class Log4netExtensions     {         public static ILoggerFactory AddLog4Net(this ILoggerFactory factory, string log4NetConfigFile)         {             factory.AddProvider(new Log4NetProvider(log4NetConfigFile));             return factory;         }         public static ILoggerFactory AddLog4Net(this ILoggerFactory factory)         {             factory.AddProvider(new Log4NetProvider("log4net.config"));             return factory;         }     } |

10. Finalmente en la clase “Startup”



11. Ejectamos la aplicacion y vemos el log, nos muestra mucho más detalle de todas las acciones que realizamos.