**Code Review**

Please review the following code snippet. Assume that all referenced assemblies have been properly included.

The code is used to log different messages throughout an application. We want the ability to be able to log to a text file, the console and/or the database. Messages can be marked as message, warning or error. We also want the ability to selectively be able to choose what gets logged, such as to be able to log only errors or only errors and warnings.

1. If you were to review the following code, what feedback would you give? Please be specific and indicate any errors that would occur as well as other best practices and code refactoring that should be done.
2. Rewrite the code based on the feedback you provided in question 1. Please include unit tests on your code.

1)

Notas:

* En verde mis comentarios y mis sugerencias en cuanto a la lógica.
* En amarillo resalto errores.
* Mejoras sugeridas:
  + Si bien la funcionalidad planteada es sencilla, se puede sugerir utilizar patrones de comportamiento, estructurales y creacionales para, por ejemplo, desacoplar el negocio de la base de datos, utilizar una fachada para simplificar la interfaz o utilizar un modelo vista/controlador.
  + Dado que el código fue copiado a un archivo PDF estimo que por eso se perdieron márgenes que el IDE de desarrollo ya establece pero igual menciono que esto es importante para la lectura del código así como la documentación del mismo.

Using System;

using System.Linq; using System.Text;

public class JobLogger

{

Private static bool \_logToFile;

Private static bool \_logToConsole;

Private static bool \_logMessage;

Private static bool \_logWarning;

Private static bool \_logError;

Private static bool LogToDatabase; /\*como buena práctica sugiero nomenclar de manera unificada los nombres de las variables\*/

Private bool \_initialized;

Public JobLogger(bool logToFile,bool logToConsole,bool logToDatabase,bool logMessage,bool logWarning,bool logError)

{

\_logError=logError;

\_logMessage=logMessage;

\_logWarning=logWarning;

LogToDatabase=logToDatabase;

\_logToFile=logToFile;

\_logToConsole=logToConsole;

\*/Mejora sugerida: las variables de \_logError, \_logMessage y \_logWarning pueden omitirse dado que no son necesarias ya que en la llamada del método LogMessage ya se define qué es lo que va a loguear o se pueden utilizar éstas variblese y omitir los parámetros message (bool), warning y error del método LogMessage. \*/

}

Public static void LogMessage(string message, bool message, bool warning, bool error) //dos parámetros con nombres iguales.

{

message.Trim(); /\*el método devuelve un string con el contendo de message quitando los caracteres en blanco, esto no se está asignando a ninguna variable\*/

if(message==null||message.Length==0) /\*Si message es null el método length va a dar error y no hay try-catch para atrapar la excepción\*/

{

return;

}

if(!\_logToConsole && !\_logToFile && !LogToDatabase)

{

Throw new Exception("Invalid configuration");

}

if((!\_logError && !\_logMessage && !\_logWarning)||(!message && !warning &&!error)) \*/Como ya mencioné anteriormente (como mejora), las variables y los parámetros que determinan qué loguear (message, error, warning) podrían unificarse, sugiero tomar las variables de clase o los parámetros de este método.\*/

{

Throw new Exception("Error or Warning or Message must be specified");

}

System.Data.SqlClient.SqlConnection connection=new

System.Data.SqlClient.SqlConnection(System.Configuration.ConfigurationManager.AppSettings["ConnectionString"]);

connection.Open(); /\*Sugiero un try-catch. Por otro lado, estas líneas deberían estar bajo la condición del guardado del log/mensaje en base (sino se abre la conexión en vano)\*/

int t;

if(message && \_logMessage)

{

t=1;

}

if(error && \_logError)

{

t=2;

}

if(warning && \_logWarning)

{

t=3;

}

/\*Sugiero agregar condiciones para determinar dónde se va a mostrar/guardar el mensaje de log (base, consola y/o file) en función de las variables LogToDatabase, \_logToFile, \_logToConsole. Como está planteado el método siempre graba el message en base, en file y se muestra por consola. \*/

System.Data.SqlClient.SqlCommand command=new

System.Data.SqlClient.SqlCommand("Insert into Log Values('"+ message +"',"+

t.ToString()+")"); /\* La variable t indica solo un número para identificar de alguna manera de qué tipo de mensaje es el log, esto se debería mejorar para mayor claridad al leer el log desde la BD. Por otro lado, se asume que la tabla está conformada por un solo dato de tipo varchar y que el ID es autoincremental (caso contrario la ejecución de esta query dará error).\*/

command.ExecuteNonQuery(); //Se sugiere try-catch

string l;

if

(!System.IO.File.Exists(System.Configuration.ConfigurationManager.AppSettings["Log FileDirectory"]+"LogFile"+DateTime.Now.ToShortDateString()+".txt"))

/\*este filelog debe existir con el nombre tal cual se menciona y se deben tener los permisos necesarios para acceder. El método devuelve true si existe el archivo pero la condición se está negando, es decir que si el archivo existe no va a ingresar al if (en donde se asigna a l todo el texto ya existente en el archivo para el día, se lleva un archivo log por día). Como está planteado, si el archivo NO existe ingresa al if y el método ReadAllText va a intentar abrir para leer el archivo y va a dar una excepción (la cual incluso no está cacheada).

Dado esto deberia ser al reves, si el archivo existe debe ingresar al if.\*/

{

l=

System.IO.File.ReadAllText(System.Configuration.ConfigurationManager.AppSettings[" LogFileDirectory"]+"LogFile"+DateTime.Now.ToShortDateString()+".txt");

/\*se sugiere cachear este método porque puede arrrojar varios tipos de excepciones, por ejemplo si se supera la cantidad de caracteres permitidos.\*/

}

if(error && \_logError)

{

l=l+DateTime.Now.ToShortDateString() + message; /\*habría que colocar algunos espacios en blanco o salto entre el texto anterior y el nuevo texto a agregar.\*/

}

if(warning && \_logWarning)

{

l=l+DateTime.Now.ToShortDateString() + message; /\*habría que colocar algunos espacios en blanco o salto entre el texto anterior y el nuevo texto a agregar.\*/

}

if(message && \_logMessage)

{

l=l+DateTime.Now.ToShortDateString() + message; /\*habría que colocar algunos espacios en blanco o salto entre el texto anterior y el nuevo texto a agregar.\*/

}

System.IO.File.WriteAllText(System.Configuration.ConfigurationManager.AppSettings[

"LogFileDirectory"]+"LogFile"+ DateTime.Now.ToShortDateString()+".txt",l);

if( error && \_logError)

{

Console.ForegroundColor=ConsoleColor.Red;

}

if( warning && \_logWarning)

{

Console.ForegroundColor=ConsoleColor.Yellow;

}

if( message && \_logMessage) /\*se asume que el parámetro message es el dato de tipo bool no el de tipo string\*/

{

Console.ForegroundColor=ConsoleColor.White;

}

Console.WriteLine(DateTime.Now.ToShortDateString() + message);

}

}

2)

Using System;

using System.Linq; using System.Text;

public class JobLogger

{

Private static bool \_logToFile;

Private static bool \_logToConsole;

Private static bool \_logMessage;

Private static bool \_logWarning;

Private static bool \_logError;

Private static bool \_logToDatabase;

Private bool \_initialized;

Public JobLogger(bool logToFile,bool logToConsole,bool logToDatabase,bool logMessage,bool logWarning,bool logError)

{

\_logError=logError;

\_logMessage=logMessage;

\_logWarning=logWarning;

\_logToDatabase=logToDatabase;

\_logToFile=logToFile;

\_logToConsole=logToConsole;

}

Public static void LogMessage(string message)

{

try

{

if(message == null)

return;

if(message.Length == 0)

return;

if(!\_logToConsole && !\_logToFile && !\_logToDatabase)

{

Throw new Exception("Invalid configuration");

}

if(!\_logError && !\_logMessage && !\_logWarning)

{

Throw new Exception("Error or Warning or Message must be specified");

}

if(\_logToConsole)

{

if(\_logError)

{

Console.ForegroundColor=ConsoleColor.Red;

}

if(\_logWarning)

{

Console.ForegroundColor=ConsoleColor.Yellow;

}

if(\_logMessage)

{

Console.ForegroundColor=ConsoleColor.White;

}

Console.WriteLine(DateTime.Now.ToShortDateString() + message.Trim());

}

if(\_logToFile)

{

string l;

if

(System.IO.File.Exists(System.Configuration.ConfigurationManager.AppSettings["Log FileDirectory"]+"LogFile"+DateTime.Now.ToShortDateString()+".txt"))

{

l=

System.IO.File.ReadAllText(System.Configuration.ConfigurationManager.AppSettings[" LogFileDirectory"]+"LogFile"+DateTime.Now.ToShortDateString()+".txt");

l= l + "\n\r" + DateTime.Now.ToShortDateString()+ “ - ” message.Trim();

System.IO.File.WriteAllText(System.Configuration.ConfigurationManager.AppSettings[

"LogFileDirectory"]+"LogFile"+ DateTime.Now.ToShortDateString()+".txt",l);

}

}

if(\_logToDatabase)

{

string t;

if(\_logMessage)

{

t= “message”;

}

if(\_logError)

{

t= “error”;

}

if(\_logWarning)

{

t= “warning”;

}

System.Data.SqlClient.SqlConnection connection=new

System.Data.SqlClient.SqlConnection(System.Configuration.ConfigurationManager.AppSettings["ConnectionString"]);

connection.Open();

System.Data.SqlClient.SqlCommand command=new

System.Data.SqlClient.SqlCommand("Insert into Log Values('"+ message.Trim() +"',"+t+")");

command.ExecuteNonQuery();

}

}

catch (Exception e)

{

throw e;

}

}

}

//unit test code

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

//también debe ir la instrucción using para poder, desde la prueba, llamar al proyecto de la app.

namespace Tests

{

[TestClass]

public class AppTests

{

[TestMethod]

public void Test1()

{

try

{

JobLogger log1 = new JobLogger (true, true, true , true, false, false);

log1.LogMessage(“Prueba de log de message”);

JobLogger log2 = new JobLogger (true, true, true , false, true, false);

log2.LogMessage(“Prueba de log de warning”);

JobLogger log3 = new JobLogger (true, true, true , false, false, true);

Log3.LogMessage(“Prueba de log de error”);

}

catch (Exception e)

{

throw e;

}

}

[TestMethod]

public void Test2()

{

try

{

JobLogger log1 = new JobLogger (true, true, false, true, false, false);

log1.LogMessage(“Prueba de log message por File y Consola”);

JobLogger log2 = new JobLogger (true, false, false, false, true, false);

log2.LogMessage(“Prueba de log warning por File”);

JobLogger log3 = new JobLogger (false, false, true , false, false, true);

Log3.LogMessage(“Prueba de log error por BD”);

}

catch (Exception e)

{

throw e;

}

}

}

}