HashLog v. 1.0 User Documentation

END USER DOCUMENTATION

Piotr Paszko

Table of Contents

Table of Contents

Project Overview

HashLog is simple and user friendly library for C# .Net Framework 4.6.1. Goal was simple: create most easy to use, runtime logging tool.

Library gets log messages from user and save it to external file and write in console. Everything is fully automatic.

License

Project is licensed by GNU v. 3 license. Feel free to use and modify! For license conditions see **LICENSE** file.

Requirements

Project is written for .NET Framework 4.6.1. Using it on other versions can make some problems.

Installation

To set up HashLog in your project follow this steps:

1. Open HashLog.sln in Visual Studio.
2. Build HashLog project with your settings.
3. Create your own project.
4. Right click on your project in Visual Studio and click **Add**->**Reference**.
5. In pop up window click **Browse** button.
6. Select builded .dll file.
7. Check HashLog reference.
8. Click **OK** button.
9. Library ready to use!

Using Example

Here’s using example of HashLog library. Simple! Isn’t it?

```
class Program
    {
        static void Main(string[] args)
        {
            //Prepare console and log file
            HashLog.HashLog.Setup("Test project");
            //Logs "Info." message with information prefix.
            HashLog.HashLog.LogInformation("Info.");
            //Logs "Warning." message with warning prefix.
            HashLog.HashLog.LogWarning("Warning.");
            //Logs "Error." message with error prefix.
            HashLog.HashLog.LogError("Error.");

            //Sends "Non-log message." without any log information (like event time or
prefix) to both outputs.
            HashLog.Output.SendBoth("Non-log message.");
            //Sends "Non-log message for console." without any log information to
console output.
            HashLog.Output.SendConsole("Non-log message for console.");
            //Sends "Non-log message for file." without any log information to console
output.
            HashLog.Output.SendFile("Non-log message for file.");


            //Prepare console and creates second log file (archive old one)
            HashLog.HashLog.Setup("Test project 2");
            //Logs "Fatal error." message with fatal error prefix and closes
application.
            HashLog.HashLog.LogFatalError("Fatal error.");
        }
    }
```

This example will create two log files. One for each `HashLog.HashLog.Setup()` calls. That method should be called once when your application is starting. `"Test project"` is name of your project. Every other line will log our messages with suitable prefixes. But `HashLog.HashLog.LogFatalError()` will also close our application due fatal error.

API Documentation

class HashLog

Class used to logging code events such as errors or doing important operations. Class will automatically add suitable prefix to log and actual system time to message.

Public Methods:

Name	Arguments	Description
`static void Setup(string projectName)`	`string projectName – name of project used in log file and console title.`	`Method is doing setup of logger. Prepare log file and console.`
`static void LogInformation(string message)`	`string message – log message.`	`Logs message with information prefix. Use when logging actual program state.`
`static void LogWarning(string message)`	`string message – log message.`	`Logs message with warning prefix. Use when logging unexpected program problems only little affecting on operations.`
`static void LogError(string message)`	`string message – log message.`	`Logs message with error prefix. Use when logging unexpected program problems affecting on operations, for example breaking it.`
`static void LogFatalError(string message)`	`string message – log message.`	`Logs message with fatal error prefix and close application. Use when logging unexpected program problems affecting on core components making whole app unable to working.`

class Output

Class used to sending messages to console and file outputs of logger in unchanged form.

Public Methods:

Name	Arguments	Description
`static void Setup(string projectName)`	`string projectName – name of project used in log file and console title.`	`Method is doing setup of logger outputs. Prepare log file and console.`
`static void SendBoth(string message)`	`string message – message text.`	`Writes message in original form from argument on both outputs.`
`static void SendConsole(string message)`	`string message – message text.`	`Writes message in original form from argument on console output only.`
`static void SendFile(string message)`	`string message – message text.`	`Writes message in original form from argument on file output only.`