BurnSystems.Logging

Purpose

The assembly BurnSystems.Logging introduces a lightweight logging tool supporting console output, debug output and a very simple internal storage for log messages.

The logger is available via a singleton “TheLog” in namespace “BurnSystems.Logging”

The assembly is a .Net Standard 2.1 assembly, so it can be used under all platforms supporting .Net Core (including Linux).

```
using BurnSystems.Logging.Provider;

namespace BurnSystems.Logging.Console
{
    class Program
    {
        static void Main(string[] args)
        {
            TheLog.AddProvider(new ConsoleProvider(), LogLevel.Info);
            TheLog.Trace("Not added");
            TheLog.Info("We have an info message.");
            TheLog.Error("Error Occured....");
            TheLog.Fatal("We have to quit the application due to loss of O².");
        }
    }
}
```

We also have included a class logger which eases the logging of events in certain classes. Sorting and Filtering can be made much easier.

```
    public class WorkingMan
    {
        private ILogger logger = new ClassLogger(typeof(WorkingMan));

        public void Work()
        {
            logger.Info("We are working.");
        }
    }
```

Usage

The following loglevels are predefined:

- Trace = 1,
- Debug = 2,
- Info = 3,
- Warn = 4,
- Error = 5,
- Fatal = 6

Providers

The following providers are supported in the first release:

1. ConsoleProvider: Each message is sent to the console
2. DebugProvider: Each message is sent to the Debug stream

3. InMemoryDatabaseProvider: A very simple database provider just being capable to store each log entry into a list. Not recommended for long running services since it will eat up your memory.
4. FileProvider: Each message will be added to a given file. During set-up of the provider, it may be decided whether the log will be extended or whether it will be cleaned up.

Filtering

Each provider and the log framework itself support the filtering of messages according to the filtering level.

1. The Global filter can be set via TheLog.FilterThreshold
2. Each provider's filter can be set during the adding of the filter:

```
TheLog.AddProvider(new ConsoleProvider(), LogLevel.Info);
TheLog.Info("Info");   // will be shown
TheLog.Trace("Trace"); // will be ignored
TheLog.Error("Error"); // will be shown
```