Logger
1.10.01

Пояснительная записка

Пт 10 Июл 2015 13:52:58

Содержание

1 Пояснительная записка проекта Logger - Журналирование многосредовых приложений

Дата

2014-2015

1.1 Аннотация

Данный документ содержит сведения о назначении динамически подключаемой библиотеки Logger - Журналирование многосредовых приложений.

Документ является Пояснительной запиской к проекту Logger и был создан утилитами Doxygen и Microsoft's HTML Help Workshop.

С утилитой Doxygen можно познакомится в документах DOXYGEN И GRAPHVIZ: ДОКУМЕНТИРОВАНИЕ ПРОЕКТОВ НА C#, БИБЛИОТЕКА РАЗБОРА АРГУМЕНТОВ КОМАНДНОЙ СТРОКИ (C#, Doxygen и Microsoft's HTML Help Workshop),

1.2 Общие сведения

Динамически подключаемая библиотека Logger написана на языке C# с использованием .Net версии 4.0. При разработке юнит-теста для тестирования библиотеки использовалась Библиотека разбора аргументов командной строки. Скачать исходный код Logger можно по адресу http://agp1.hx0.ru/Logger.html

1.3 Назначение

Назначение работы библиотеки заключается в выводе текстовых сообщений различного уровня важности в текстовый файл (журнал работы приложения), находящийся в том же каталоге, откуда запускается приложение. Если не задано противное, то журнал работы приложения имеет такое

же имя, как и само приложение, а расширение - не exe, а log. Например, Юнит-тест строится с названием app.exe, тогда его журнал будет иметь название app.log. В случае если приложение консольное, есть возможность дублировать некоторые сообщения в стандартный вывод ошибок. Кроме того, в библиотеке приводится пример наследования от интерфейса IDisposable.

1.4 Процедура установки

Постройте динамически подключаемую библиотеку и добавьте ссылку на неё в свой проект.

1.5 Процедура проверки работоспособности

Для тестирования библиотеки используется специальное приложение - Юнит-тест. Находящийся в корневом каталоге проекта Файл test.cmd содержит примеры вызова Юнит-тест. Если в результате исполнения файла test.cmd в окне консоли появится текст похожий на следующий:

```
app.exe -v -?
to demo 2 threads with Monitor class
usage:
app [-?] [-d] [-v] [-l LLL] [-ln NNN] [-s SSS] [-m MAX]  ...
options:
  -?              : to see this help: True
  -d              : debug mode: False
  -v              : additional info: True
  -l LLL        : log level (1..8): 1
  -ln NNN       : log level name {Spam Debug Warning Stats Error FatalError Info Ignore}: 'Ignore'
  -s SSS        : msecs to sleep: 125
  -m MAX        : to count prime numbers up to MAX (1..): 1000
'?' means the same as 'help'
'd' means the same as 'debug'
'v' means the same as 'verbose'
'l' means the same as 'log'
'ln' means the same as 'logName'
's' means the same as 'sleep'
'm' means the same as 'max'
app.exe -d -l 10  -v
[08.07.2015 18:55:21]: [Info]       first: next prime is 3
[08.07.2015 18:55:21]: [Info]       first: next prime is 5
[08.07.2015 18:55:21]: [Info]       second: next prime is 7
[08.07.2015 18:55:22]: [Info]       second: next prime is 11
[08.07.2015 18:55:22]: [Info]       second: next prime is 13
[08.07.2015 18:55:22]: [Info]       second: next prime is 17
[08.07.2015 18:55:22]: [Info]       second: next prime is 19
[08.07.2015 18:55:23]: [Info]       first: next prime is 23
[08.07.2015 18:55:23]: [Info]       first: next prime is 29
[08.07.2015 18:55:23]: [Info]       first: next prime is 31
[08.07.2015 18:55:23]: [Info]       first: next prime is 37
[08.07.2015 18:55:24]: [Info]       second: next prime is 41
[08.07.2015 18:55:24]: [Info]       second: next prime is 43
[08.07.2015 18:55:24]: [Info]       second: next prime is 47
[08.07.2015 18:55:25]: [Info]       first: next prime is 53
[08.07.2015 18:55:25]: [Info]       second: next prime is 59
[08.07.2015 18:55:25]: [Info]       second: next prime is 61
[08.07.2015 18:55:26]: [Info]       first: next prime is 67
[08.07.2015 18:55:26]: [Info]       first: next prime is 71
[08.07.2015 18:55:26]: [Info]       first: next prime is 73
[08.07.2015 18:55:26]: [Info]       first: next prime is 79
[08.07.2015 18:55:27]: [Info]       second: next prime is 83
[08.07.2015 18:55:27]: [Info]       first: next prime is 89
[08.07.2015 18:55:28]: [Info]       second: next prime is 97
[08.07.2015 18:55:28]: [Stats]      thread 'first' finished with 53 numbers
[08.07.2015 18:55:28]: [Info]       second: next prime is 101
[08.07.2015 18:55:28]: [Stats]      thread 'second' finished with 44 numbers
[08.07.2015 18:55:28]: [Stats]      time of work is 7,203125 secs
```

то тестирование Logger можно считать успешным.

Содержимое командного файла test.cmd:

```
echo app.exe -v -?
app.exe -v -?
echo app.exe -d -l 10  -v
app.exe -m 100 -d -l 10  -v
```

1.6 Использованные параметры и команды Doxygen-a

При создании документа были использованы следующие параметры файла конфигурации, команды форматирования текста и синонимы для них (в оригинале markdown), отличные от описанных в DOXYGEN И GRAPHVIZ : ДОКУМЕНТИРОВАНИЕ ПРОЕКТОВ НА C#, БИБЛИОТЕКА РАЗБОРА АРГУМЕНТОВ КОМАНДНОЙ СТРОКИ (C#, Doxygen и Microsoft's HTML Help Workshop):

- COMPACT_LATEX = YES. Видимых отличий в тексте сгенерированных с различными значениями этого параметра заметить не удалось, но размер выходной pdf-файл при включенном параметре оказывался на четверть меньше.
- ALIASES = nm=Logger. Использование параметра позволяет использовать команду \nm вместо явного упоминания названия Logger.
- '\section'. Команда создает новый раздел документа, после слова section сначала пишется тег для внутренних ссылок, а потом заколовок раздела.
- '\verbinclude' - содержимое указанного файла выводит без форматирования. Этой командой в документацию было выведено содержимое файла test.cmd. Каталог в котором команда производит поиск задается в параметре EXAMPLE_PATH.
- EXAMPLE_PATH = .. Это каталог с файлами для команды '\verbinclude'.
- '\verbatim', '\endverbatim'. '\verbatim' - отключает форматирование текста, '\endverbatim' - включает форматирование.
- '\tableofcontents'. Команда приводит к некрасивому выводу в pdf-файле, поэтому не использовалась.

1.7 Замеченные проблемы

По необнаруженным причинам иногда вывод в журнал продолжается после закрытия файла. До сих пор не было времени пофиксить проблему. Оставленные до лучших времен примеры приложений с этой проблемой на момент создания документации отказались выбрасывать эксепшены и работают хорошо. :-((

1.8 Динамическая загрузка Logger

По работе пришлось написать пример использования системы отражения (System.Reflection) для динамической загрузки библиотеки (то есть, на шаге исполнения, без указания линкеру где находится сборка Logger). Командой diff (утилита юникса для получения разницы файлов):

diff Program.cs ../reflection_dynamic_load/Program.cs

был получен следующий файл с разницей обоих юнит-тестов:

```
0a1,2
> ///https://msdn.microsoft.com/en-us/library/d133hta4.aspx
>
2a5,6
>
>
3a8
> using System.Runtime.Remoting;
10c15,21
```

```
< using Logger;
---
>
> // пример динамической загрузки библиотеки журналирования
> /// в принципе пространство можно оставить я же его знаю все равно
> //using Logger;
>
> using System.Reflection;
>
23,26c34,41
< /*        static  Program(){
<           var format = new System.Globalization.NumberFormatInfo();
<           format.NumberDecimalSeparator = ".";
<         }  */
---
>         static void Types(Assembly ass)
>         {
>             Console.WriteLine("Типы сборки {0}: \n",ass.FullName);
>             Type[] types = ass.GetTypes();
>             foreach (Type t in types)
>                 Console.WriteLine("--> " + t);
>             Console.WriteLine();
>         }
27a43
>         static public  Type ImpLevel;
31c47
<         static public ArgIntMM     logLvl ; ///<чтобы задать уровень журналирования числом.
---
> //        static public ArgIntMM     logLvl ; ///<чтобы задать уровень журналирования числом.
38c54
<             string lLvl = "log level names:{"+Loger.ILList()+"}";
---
>             string lLvl = "log level names:{ Spam, Error, Debug, Warning }";
42,43c58
<             logLvl =  new ArgIntMM(1,    "l",  "log",  "log level", "LLL");
<             logNm  =  new ArgStr  ("Error",    "ln",  "logName",   lLvl, "NNN");
---
>             logNm  =  new ArgStr  ("Error",    "l",  "log",   lLvl, "NNN");
45,46c60,62
<             logLvl.setMin(1);
<             logLvl.setMax(8);
---
> //            logLvl =  new ArgIntMM(1,    "l",  "log",  "log level", "LLL");
> //            logLvl.setMin(1);
> //            logLvl.setMax(8);
83d98
<             IMPORTANCELEVEL x = IMPORTANCELEVEL.Error;
92,93d106
<               else if (logLvl.check(ref i, args))
<                 ;
95c108
<                     x = Loger.strtoLvl(logNm);
---
>                     ;
103,108c116,146
<             using (Loger l = new Loger(logNm, dbgF)){
<               if (vF)
<                 l.cnslLvl = IMPORTANCELEVEL.Stats;
<               primer a = new primer("first", l, ThreadPriority.Normal);
<               primer b = new primer("second", l);
<       //        a.t.Priority = ThreadPriority.Lowest;
---
>
> /// использование отражения -------------------------
>
>
> Assembly assembly = Assembly.LoadFrom("Logger.cs.dll");  // загрузили длл
> Types(assembly);
> Type type = assembly.GetType("Logger.Loger");             // взяли тип Логер
> ImpLevel = assembly.GetType("Logger.IMPORTANCELEVEL");   // взяли тип уровень важности
>
>
> //http://www.sql.ru/forum/548476/methodinfo-invoke-dlya-peregruzhennyh-metodov
```

```
> //   Type t = Type.GetType("TestForUserControls.Form2");
> //
>    string iLvl = logNm;
>    object ol = type.GetConstructor(new Type[] { typeof(string)}).Invoke(new object[] { iLvl});
>     // это вызов конструктора  new Loger(logNm) - с заданным уровнем журналирования
> ///object ol = Activator.CreateInstance(type);      //  вызвать Loger() получилось, специально делал такой конструктор
> ///object ol = Activator.CreateInstance(type, logNm); // так не получилось использовать Loger(logNm)
>                                                     // , видиом надо явно к строке преобразовать.
> //            using (Loger l = new Loger(logNm, dbgF)) ---------- отражение заменяет этот оператор
>
>            {
>              if (vF)  {                    ///-------------------
> MethodInfo method = type.GetMethod("setCnslLvl"); ///  задать уровень вывода журнала в консоль
> object[] pars = new object[1];
> pars[0] =  "Spam";
> Object retVal = method.Invoke(ol, pars);    ///-------------------
>              }
>              primer a = new primer("first", ol,  ThreadPriority.Normal);
>              primer b = new primer("second", ol );
>              a.t.Priority = ThreadPriority.Lowest;
110,111c148,150
<              l.WriteLine(IMPORTANCELEVEL.Stats,"thread '{0}' finished with {1}/{2} numbers/primers"
<                      , "first", a.numbers, a.primers);
---
>         /// эти выводы удалил лень оформалять вызов WriteLine
>         ///      l.WriteLine(IMPORTANCELEVEL.Stats,"thread '{0}' finished with {1}/{2} numbers/primers"
>         ///              , "first", a.numbers, a.primers);
113,114c152,153
<              l.WriteLine(IMPORTANCELEVEL.Stats,"thread '{0}' finished with {1}/{2} numbers/primers"
<                      , "second", b.numbers, b.primers);
---
>         ///      l.WriteLine(IMPORTANCELEVEL.Stats,"thread '{0}' finished with {1}/{2} numbers/primers"
>         ///              , "second", b.numbers, b.primers);
116,117c155,156
<              l.WriteLine(IMPORTANCELEVEL.Stats, "time of work is {0} secs"
<                  , (fn - st).TotalSeconds);
---
>          ///     l.WriteLine(IMPORTANCELEVEL.Stats, "time of work is {0} secs"
>          ///         , (fn - st).TotalSeconds);
119a159
> ((IDisposable)ol).Dispose();          /// юзинга нет, надо явно вызвать диспоуз
130c170
<       Loger log = null;
---
>       object log = null;
133c173
<                  , Loger l     ///<журнал для вывода сообщений;
---
>                  , object l     ///<журнал для вывода сообщений;
155c195,212
<         while (i < Program.max){
---
> //---------------------------------------- взяли ссылку на writeLine
> Assembly assembly = Assembly.LoadFrom("Logger.cs.dll");
> Type t = assembly.GetType("Logger.Loger");
> /*
> MethodInfo method = t.GetMethod("WriteLine",  new Type[] { typeof(string)});
> */
> MethodInfo method = t.GetMethod("WriteLine",
>   BindingFlags.Public | BindingFlags.Instance,
>   null,
>   new [] {  Program.ImpLevel, typeof(string), typeof(object[])},
>   null);
> //----------------------------------------
> if (method == null)
>         Console.WriteLine("{0}: there is no your method", name);
> else
>         Console.WriteLine("{0}: there is  your method!", name);
>
> //----------------------------------------
156a214,215
>
>         while (i < Program.max){
```

```
166,169d224
< //          lock (typeof(Program)){
< //            numbers++;
< //            i = Program.current++;
< //          }
179a235,236
>
>
182,185c239,264
<              log.WriteLine(IMPORTANCELEVEL.Info,"{0}: next prime is {1}", name, i);
<            }
<            else
<              log.WriteLine(IMPORTANCELEVEL.Debug,"{0}: {1} is not a prime", name, i);
---
> //             log.WriteLine(IMPORTANCELEVEL.Info,"{0}: next prime is {1}", name, i);
> //---------------------------- сообщение о простом числе, уровень важности Info
>              object[] pars = new object[3];
>                object[] pars2 =  new object[2];
>                pars2[0]   = name;
>                pars2[1]   = i;
>              pars[0] =  6 ;  //инфо
> //              pars[0] =
>              pars[1] =  "{0}: next prime is {1}";
>              pars[2] = pars2;
>
>              Object retVal = method.Invoke(log, pars);
> //---------------------------------------
>            }
>            else  {
> //------------------------------ сообщение о не простом числе, уровень важности Debug
>              object[] pars = new object[3];
>              pars[0] =  1 ;  // дебаг
>              pars[1] = "{0}: {1} is not a prime";
>               object[] pars2 =  new object[2];
>               pars2[0]   = name;
>               pars2[1]   = i;
>              pars[2] = pars2;
>              Object retVal = method.Invoke(log, pars);
> //---------------------------------------
>            }
```

Замечательный пример, демонстрирующий существенное повышение трудоемкости и ненужности, без острой необходимости, такого программирования.

2 Алфавитный указатель пространств имен

2.1 Пространства имен

Полный список пространств имен.

3 Иерархический список классов

3.1 Иерархия классов

Иерархия классов.

IDisposable

4 Алфавитный указатель классов

4.1 Классы

Классы с их кратким описанием.

5 Список файлов

5.1 Файлы

Полный список файлов.

6 Пространства имен

6.1 Пакет Logger

Именованная область видимости библитеки Журналирование.

Классы

- class Loger
  синоним для класса LOGGER.
- class LOGGER
- class pair

вспомогательный класс. Пара (уровень важности, сообщение) ставится в очередь сообщений.

Перечисления

- enum IMPORTANCELEVEL { IMPORTANCELEVEL.Spam, IMPORTANCELEVEL.Debug, IMPORTANCELEVEL.Warning, IMPORTANCELEVEL.Stats, IMPORTANCELEVEL.Error, IMPORTANCELEVEL.FatalError, IMPORTANCELEVEL.Info, IMPORTANCELEVEL.Ignore }

  Перечисление уровней важности сообщения

6.1.1 Подробное описание

Именованная область видимости библитеки Журналирование.

6.1.2 Перечисления

6.1.2.1 enum Logger.IMPORTANCELEVEL

Перечисление уровней важности сообщения

Элементы перечислений

Spam мусорные сообщения

Debug отладка

Warning предупреждения

Stats менне важная информация

Error ошибки приложения

FatalError катастрофические ошибки, делающие навозможным функционирвание приложения

Info очень важная информация

Ignore для отсутствия вывода вообще

6.2 Пакет test

Именованная область видимости для тестирования Logger.

Классы

- class primer

  подсчет простых чисел в отдельном среде.
- class Program

  содержит точку входа Main в юнит-тест. Кроме этого файл содержит глобальные переменные для управления работой юнит-теста.

6.2.1 Подробное описание

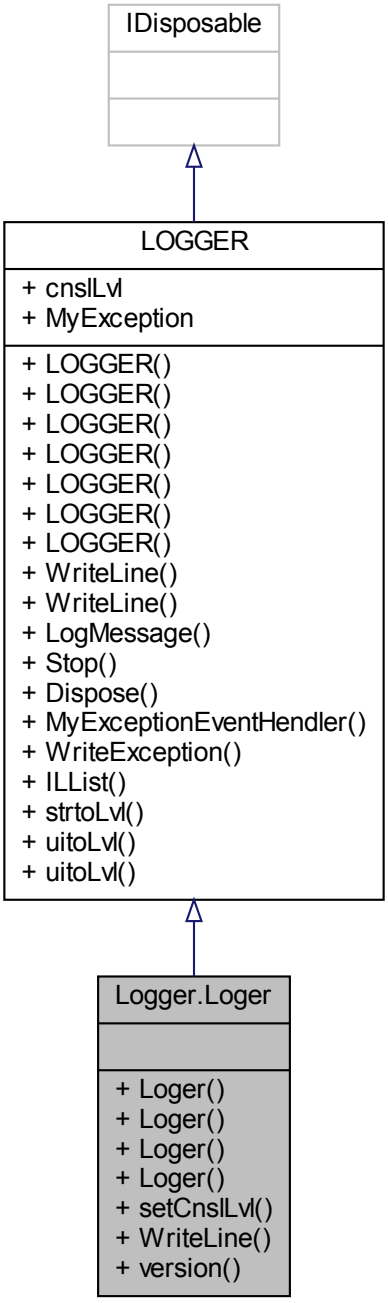
Именованная область видимости для тестирования Logger.

7 Классы

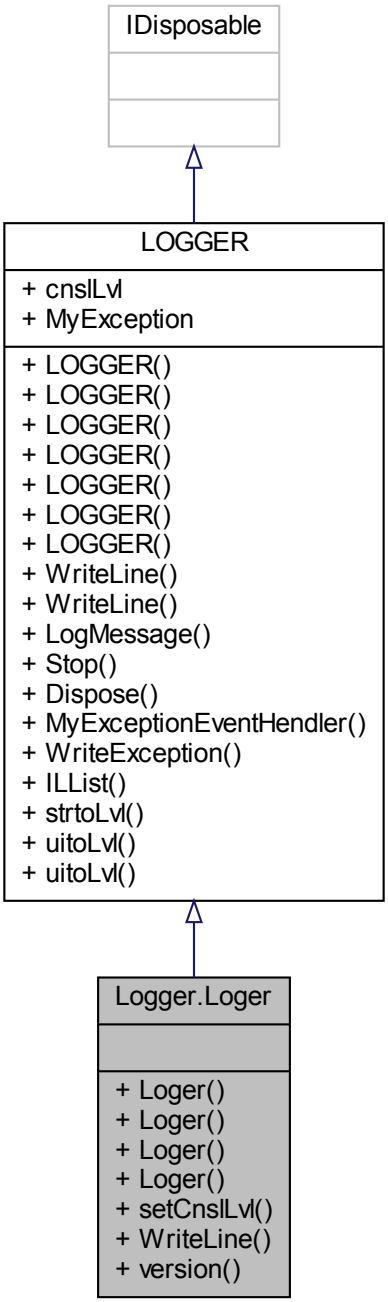
7.1 Класс Logger.Loger

синоним для класса LOGGER.

Граф наследования:Logger.Loger:



Граф связей класса Logger.Loger:



Открытые члены

- Loger (IMPORTANCELEVEL ImportanceLevel, bool lDbg, string flNm)
- Loger ()
- Loger (string impLevel)
- Loger (string impLevel, bool lDbg)
- void setCnslLvl (string lvl)
- void WriteLine (String Format)

Открытые статические члены

- static void version (out int major, out int minor, out int build)

  версия библиотеки

Дополнительные унаследованные члены

7.1.1 Подробное описание

синоним для класса LOGGER.

Лично мне не нравятся названия классов записанные большими буквами, но уже написано слишком много проектов с журналированием, что бы можно было заменить название. Сообщения для журналирования не выводятся в файл журнала, а ставятся в очередь сообщений. Из очереди сообщений в файл их выводит специальный сред, который работает с самым низким приоритетом - ThreadPriority.Lowest и засыпает Thread.Sleep(10) при отсутствии сообщений в очереди.

7.1.2 Конструктор(ы)

7.1.2.1 Logger.Loger.Loger (IMPORTANCELEVEL ImportanceLevel, bool lDbg, string flNm) [inline]

В конструкторе главный параметр уровень важности При помощи него можно задать какие сообщения будут сохраняться в журнале, а какие - нет. Если явно не задается имя журнала, то оно будет совпадать с именем приложения, за исключением расширения, оно будет не exe, а log.

Аргументы

ImportanceLevel	минимальный уровень важности сохраняемых сообщений;
lDbg	отладка библиотеки журналирования, если установлен в true, то файл журнала закрывается после каждого вывода. Приводит к существенному замедлению работы журналирования;
flNm	имя файла журнала.

7.1.2.2 Logger.Loger.Loger ( ) [inline]

7.1.2.3 Logger.Loger.Loger (string impLevel) [inline]

Аргументы

impLevel	минимальный уровень важности как текст, при неправильно заданном уровне будет установлен в Error

7.1.2.4 Logger.Loger.Loger (string impLevel, bool lDbg) [inline]

Аргументы

impLevel	минимальный уровень важности, как текст

7.1.3 Методы

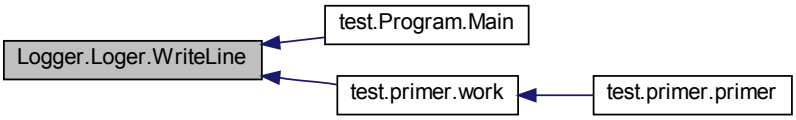
7.1.3.1 void Logger.Loger.setCnslLvl (string lvl) [inline]

7.1.3.2 static void Logger.Loger.version (out int major, out int minor, out int build) [inline], [static]

версия библиотеки

7.1.3.3 void Logger.Loger.WriteLine (String Format) [inline]

Граф вызова функции:

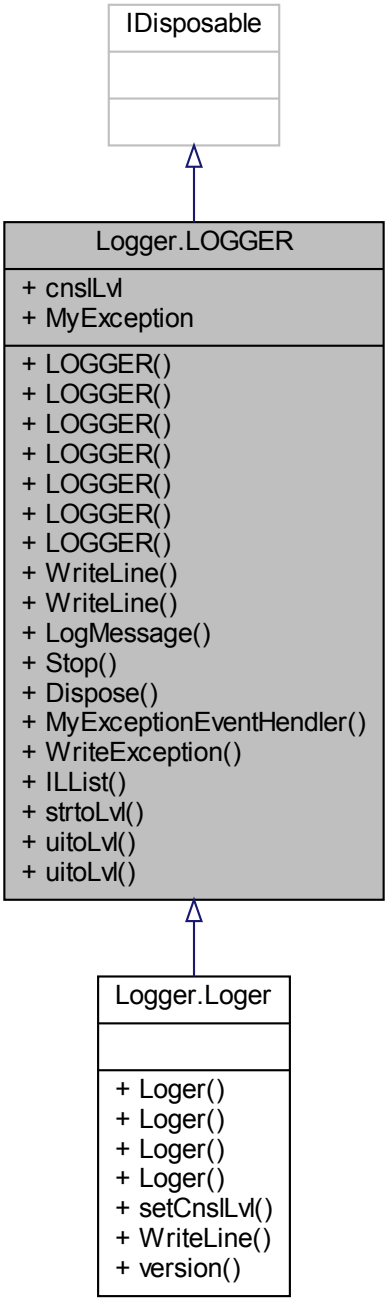


Объявления и описания членов класса находятся в файле:

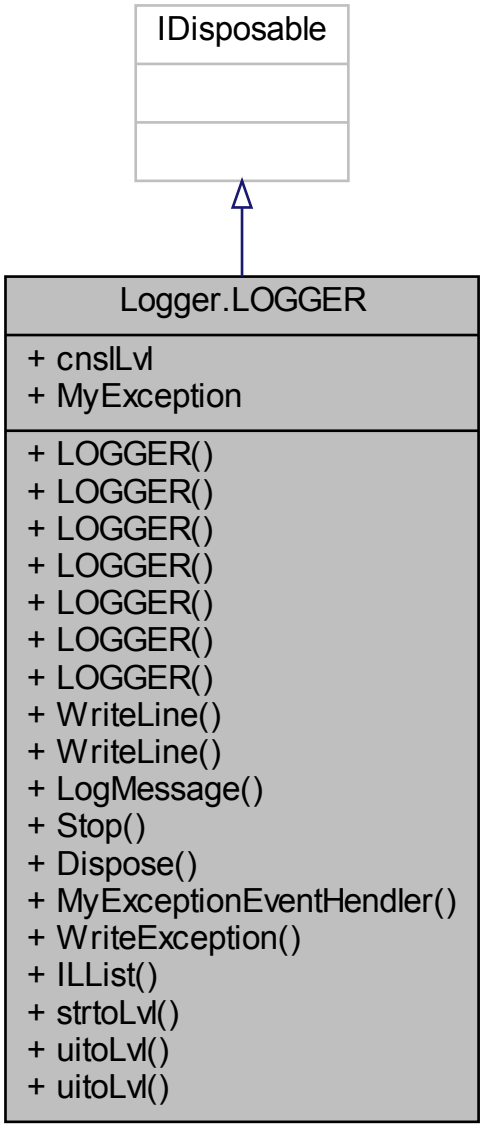
- cs/Logger.cs

7.2 Класс Logger.LOGGER

Граф наследования:Logger.LOGGER:



Граф связей класса Logger.LOGGER:



Открытые члены

- LOGGER (string impLevel, bool lDbg, string fn)

  Констурктор, создаст и запустит логер
- LOGGER (IMPORTANCELEVEL ImportanceLevel, bool lDbg, string fn)
- LOGGER (uint lvl)

  устарел
- LOGGER (int lvl)

  устарел
- LOGGER (IMPORTANCELEVEL ImportanceLevel)
- LOGGER (IMPORTANCELEVEL ImportanceLevel, string flNm)

- LOGGER (IMPORTANCELEVEL ImportanceLevel, bool lDbg)
- void WriteLine (String Format, params object[] Segments)

  для удобства заменты Console.WriteLine() на Loger.WriteLine()
- void WriteLine (IMPORTANCELEVEL Importance, String Format, params object[] Segments)

  Метод-аналог Console.WriteLine() с пользовательскими параметрами
- void LogMessage ()

  метод вывода сообщений в файл журнала. Работает в отдельном потоке.
- void Stop ()

  Метод остановки логера и его среда
- void Dispose ()

  Метод остановки логера и его среда
- delegate void MyExceptionEventHendler (object sender, Exception e)
- void WriteException (Exception e)

Открытые статические члены

- static string ILList ()

  метод выдает список уровней важности для подсказки оператору.
- static IMPORTANCELEVEL strtoLvl (string code)
- static IMPORTANCELEVEL uitoLvl (int code)
- static IMPORTANCELEVEL uitoLvl (uint code)

Открытые атрибуты

- IMPORTANCELEVEL cnslLvl = IMPORTANCELEVEL.Ignore

  дублирование вывода в консоль, менее важные сообщения будут игнорироваться
- MyExceptionEventHendler MyException

7.2.1 Конструктор(ы)

7.2.1.1 Logger.LOGGER.LOGGER (string impLevel, bool lDbg, string fn) [inline]

Констурктор, создаст и запустит логер

7.2.1.2 Logger.LOGGER.LOGGER (IMPORTANCELEVEL ImportanceLevel, bool lDbg, string fn) [inline]

7.2.1.3 Logger.LOGGER.LOGGER (uint lvl) [inline]

устарел

7.2.1.4 Logger.LOGGER.LOGGER (int lvl) [inline]

устарел

7.2.1.5 Logger.LOGGER.LOGGER (IMPORTANCELEVEL ImportanceLevel) [inline]

7.2.1.6 Logger.LOGGER.LOGGER (IMPORTANCELEVEL ImportanceLevel, string flNm) [inline]

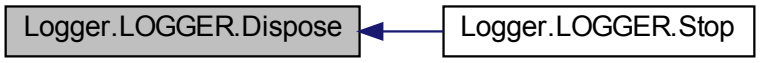
7.2.1.7 Logger.LOGGER.LOGGER (IMPORTANCELEVEL ImportanceLevel, bool lDbg) [inline]

7.2.2 Методы

7.2.2.1 void Logger.LOGGER.Dispose ( ) [inline]

Метод остановки логера и его среда

Граф вызова функции:



7.2.2.2 static string Logger.LOGGER.ILList ( ) [inline], [static]

метод выдает список уровней важности для подсказки оператору.

7.2.2.3 void Logger.LOGGER.LogMessage ( ) [inline]

метод вывода сообщений в файл журнала. Работает в отдельном потоке.

Нет особой нужды, чтоб этот метод был публичным Для блокирования очереди сообщений используется оператор lock (this){ } В случае отсутствия сообщений в очереди, сред засыпает на некоторое время.

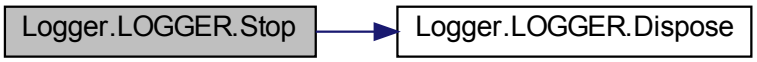
7.2.2.4 delegate void Logger.LOGGER.MyExceptionEventHendler (object sender, Exception e)

7.2.2.5 void Logger.LOGGER.Stop ( ) [inline]

Метод остановки логера и его среда

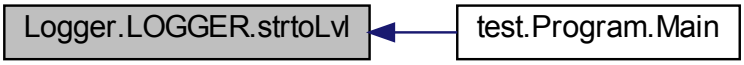
Стандартное название функции деструктор, освобождает важные ресурсы, которые не относяться к памяти

Граф вызовов:



7.2.2.6 static IMPORTANCELEVEL Logger.LOGGER.strtoLvl (string code) [inline], [static]

Граф вызова функции:

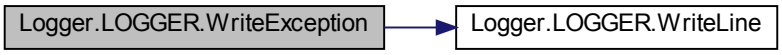


7.2.2.7 static IMPORTANCELEVEL Logger.LOGGER.uitoLvl (int code) [inline], [static]

7.2.2.8 static IMPORTANCELEVEL Logger.LOGGER.uitoLvl (uint code) [inline], [static]

7.2.2.9 void Logger.LOGGER.WriteException (Exception e) [inline]

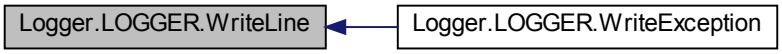
Граф вызовов:



7.2.2.10 void Logger.LOGGER.WriteLine (String Format, params object[] Segments) [inline]

для удобства заменты Console.WriteLine() на Loger.WriteLine()

Граф вызова функции:



7.2.2.11 void Logger.LOGGER.WriteLine (IMPORTANCELEVEL Importance, String Format, params object[] Segments) [inline]

Метод-аналог Console.WriteLine() с пользовательскими параметрами

Метод блокирует доступ к журналу оператором lock (this) {}, поэтому чтение из очереди сообщений (LogMessage()) и поставка в очередь выполняются по очереди

Аргументы

Importance	важность данного сообщения
Format	строка с форматом сообщения
Segments	параметры сообщения

7.2.3 Данные класса

7.2.3.1 IMPORTANCELEVEL Logger.LOGGER.cnslLvl = IMPORTANCELEVEL.Ignore

дублирование вывода в консоль, менее важные сообщения будут игнорироваться

7.2.3.2 MyExceptionEventHendler Logger.LOGGER.MyException

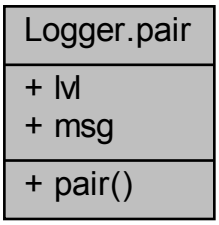
Объявления и описания членов класса находятся в файле:

- cs/Logger.cs

7.3 Класс Logger.pair

вспомогательный класс. Пара (уровень важности, сообщение) ставится в очередь сообщений.

Граф связей класса Logger.pair:



Открытые члены

- pair (IMPORTANCELEVEL l, string m)

Открытые атрибуты

- IMPORTANCELEVEL lvl
- string msg

7.3.1 Подробное описание

вспомогательный класс. Пара (уровень важности, сообщение) ставится в очередь сообщений.

7.3.2 Конструктор(ы)

7.3.2.1 Logger.pair.pair (IMPORTANCELEVEL l, string m) [inline]

7.3.3 Данные класса

7.3.3.1 IMPORTANCELEVEL Logger.pair.lvl

7.3.3.2 string Logger.pair.msg

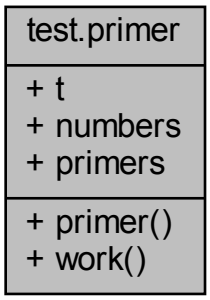
Объявления и описания членов класса находятся в файле:

- cs/Logger.cs

7.4 Класс test.primer

подсчет простых чисел в отдельном среде.

Граф связей класса test.primer:



Открытые члены

- primer (string nm, Loger l, ThreadPriority p=ThreadPriority.Lowest)
- void work (object o)

  функция, выполняющийся в среде. Фукнция увеличивает статическую переменную Program.current. Чтобы гарантировать строгую очередность увеличения переменной используется класс Monitor, метод Monitor.Enter(typeof(Program)) которого блокирует доступ к статическим переменным Program, а метод Monitor.Exit(typeof(Program)) разблокирует и, значит, дает доступ к этим переменным другому среду.

Открытые атрибуты

- Thread t = null
- int numbers = 0

  количество целых чисел, проверенных средом
- int primers = 0

  количество простых чисел, полученных средом

7.4.1 Подробное описание

подсчет простых чисел в отдельном среде.

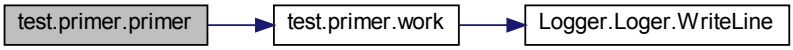
7.4.2 Конструктор(ы)

7.4.2.1 test.primer.primer (string nm, Loger l, ThreadPriority p = ThreadPriority.Lowest) [inline]

Аргументы

nm	название среда;
l	журнал для вывода сообщений;

Граф вызовов:

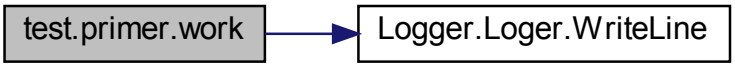


7.4.3 Методы

7.4.3.1 void test.primer.work (object o) [inline]

функция, выполняющийся в среде. Фукнция увеличивает статическую переменную Program.current. Чтобы гарантировать строгую очередность увеличения переменной используется класс Monitor, метод Monitor.Enter(typeof(Program)) которого блокирует доступ к статическим переменным Program, а метод Monitor.Exit(typeof(Program)) разблокирует и, значит, дает доступ к этим переменным другому среду.

Граф вызовов:



Граф вызова функции:

7.4.4 Данные класса

7.4.4.1 int test.primer.numbers = 0

количество целых чисел, проверенных средом

7.4.4.2 int test.primer.primers = 0

количество простых чисел, полученных средом

7.4.4.3 Thread test.primer.t = null

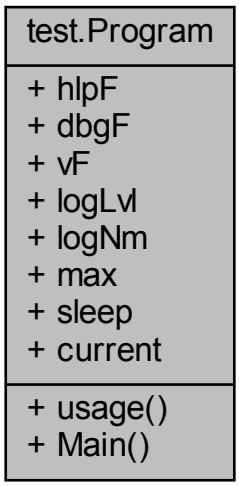
Объявления и описания членов класса находятся в файле:

- Program.cs

7.5 Класс test.Program

содержит точку входа Main в юнит-тест. Кроме этого файл содержит глобальные переменные для управления работой юнит-теста.

Граф связей класса test.Program:



Открытые статические члены

- static void usage ()

  программа выдачи подсказки по использованию юнит-теста.
- static void Main (string[] args)

  Метод для тестирования Logger. Класс Loger является наследником интерфейса IDisposable и поэтому может вызываться с использованием специального оператора using (Loger l = new Loger(logLvl)){}, который гарантирует закрытие журналирования при помощи метода Dispose(). В методе создаются два объекта типа primer - a и b , запускающих различных среда primer.t для подсчета простых чисел, которые записывают в журнал сообщения о найденных простых числах (с уровнем важности Info) и сообщения о непростых числах (с уровнем важности Debug). Работа главного среда блокируется операторами a.t.Join();, до завершения выполняющихся средов primer.t.

Статические открытые данные

- static ArgFlg hlpF

  выдать подсказку юнит-теста.
- static ArgFlg dbgF

  открывать-закрывать журнал при каждом выводе сообщения
- static ArgFlg vF

  дополнительный вывод в консоль.
- static ArgIntMM logLvl

чтобы задать уровень журналирования числом.

- static ArgStr logNm

  чтобы задать уровень журналирования именем (в данном приложении не используется).

- static ArgIntMM max

  максимальное целое, которое будем проверять на простоту.

- static ArgInt sleep

  милисекунды для засыпания среда, вычисляющего простые числа.

- static int current = 3

  текущий претендент на выполнение свойства быть простым числом

7.5.1 Подробное описание

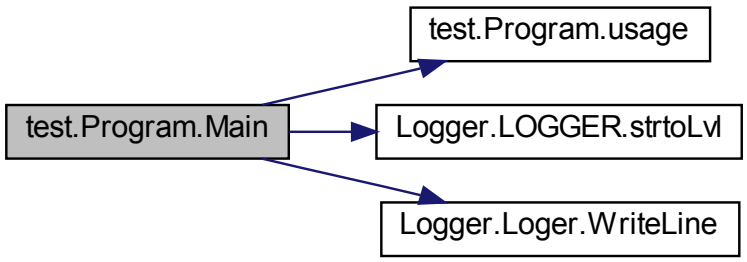
содержит точку входа Main в юнит-тест. Кроме этого файл содержит глобальные переменные для управления работой юнит-теста.

7.5.2 Методы

7.5.2.1 static void test.Program.Main (string[] args) [inline], [static]

Метод для тестирования Logger. Класс Loger является наследником интерфейса IDisposable и поэтому может вызываться с использованием специального оператора using (Loger l = new Loger(log↩ Lvl)){}, который гарантирует закрытие журналирования при помощи метода Dispose(). В методе создаются два объекта типа primer - a и b , запускающих различных среда primer.t для подсчета простых чисел, которые записывают в журнал сообщения о найденных простых числах (с уровнем важности Info) и сообщения о непростых числах (с уровнем важности Debug). Работа главного среда блокируется операторами a.t.Join();, до завершения выполняющихся средов primer.t.

Граф вызовов:



7.5.2.2 static void test.Program.usage ( ) [inline], [static]

программа выдачи подсказки по использованию юнит-теста.

Граф вызова функции:

7.5.3 Данные класса

7.5.3.1 int test.Program.current = 3 [static]

текущий претендент на выполнение свойства быть простым числом

7.5.3.2 ArgFlg test.Program.dbgF [static]

открывать-закрывать журнал при каждом выводе сообщения

7.5.3.3 ArgFlg test.Program.hlpF [static]

выдать подсказку юнит-теста.

7.5.3.4 ArgIntMM test.Program.logLvl [static]

чтобы задать уровень журналирования числом.

7.5.3.5 ArgStr test.Program.logNm [static]

чтобы задать уровень журналирования именем (в данном приложении не используется).

7.5.3.6 ArgIntMM test.Program.max [static]

максимальное целое, которое будем проверять на простоту.

7.5.3.7 ArgInt test.Program.sleep [static]

милисекунды для засыпания среда, вычисляющего простые числа.

7.5.3.8 ArgFlg test.Program.vF [static]

дополнительный вывод в консоль.

Объявления и описания членов класса находятся в файле:

- Program.cs

8 Файлы

8.1 Файл cs/Logger.cs

Классы

- class Logger.pair

  вспомогательный класс. Пара (уровень важности, сообщение) ставится в очередь сообщений.

- class Logger.Loger

  синоним для класса LOGGER.
- class Logger.LOGGER

Пространства имен

- package Logger

  Именованная область видимости библитеки Журналирование.

Перечисления

- enum Logger.IMPORTANCELEVEL { Logger.IMPORTANCELEVEL.Spam, Logger.IMPORTANCELEVEL.Debug, Logger.IMPORTANCELEVEL.Warning, Logger.IMPORTANCELEVEL.Stats, Logger.IMPORTANCELEVEL.Error, Logger.IMPORTANCELEVEL.FatalError, Logger.IMPORTANCELEVEL.Info, Logger.IMPORTANCELEVEL.Ignore }

  Перечисление уровней важности сообщения

8.2 Файл cs/Logger.txt

8.3 Файл Program.cs

Классы

- class test.Program

  содержит точку входа Main в юнит-тест. Кроме этого файл содержит глобальные переменные для управления работой юнит-теста.
- class test.primer

  подсчет простых чисел в отдельном среде.

Пространства имен

- package test

  Именованная область видимости для тестирования Logger.

Предметный указатель