Logger: Руководство программиста

DocumentId:GradSoft-PR-09.08.2000-v1.2.0

May 8, 2003

1 Введение

Logger представляет собой компоненту для организации вывода сообщений в log файл и организации вызова пользовательских функций по событиям.

Этот документ представляет собой неформальное описание, полная спецификация пакета приводится в API reference (`www.gradsoft.kiev.ua/common/ToolBox/Logger/API/index.html`).

2 Общее описание механизма работы

Процесс работы пользовательского приложения с Logger выглядит следующим образом:

- Пользователь создает объект типа Logger, указывая там основные параметры.
- Объект типа Logger предоставляет виртуальные потоки вывода для каждого из типов сообщений. Предопределенны 5 типов сообщений: Debug, Info, Warning, Errors, Fatals.
- При необходимости выполнения определенных действий, при наступлении событий из набора определенных типов, вызывает процедуру Logger::setCallback
- Для записи потока пользователь использует аттрибуты:
  - Logger::debugs(),
  - Logger::infos(),
  - Logger::warnings(),
  - Logger::errors(),
  - Logger::fatals(),

  следующим образом:

```
logger.errors() << "This is error:" << 334 << endl;
```

После выполнения этой строки в log файл будет записана строка `"This is error:334"` и если была установленна callback функция для ошибок, то она будет вызванна с аргументом `"This is error:334"`

2.1 Конфигурирование времени компиляции

Возможно отключать (или включать) запись в потоки вывода во время компиляции, определяя соответствующие препроцессорные символы в true или false:

- `LOG_DEBUG_ENABLE` - включить вывод в поток отладки. Когда `LOG_DEBUG_ENABLE` установленно в `true`, выражение `logger.debugs() << msg << endl;` выполняется так, как было описанно выше. В противном случае это выражение на этапе компиляции редуцируется в предложение, не выполняющее никаких операций. По умолчанию этот символ установлен в `false`.
- `LOG_INFO_ENABLE` - включить вывод в информационный поток. По умолчанию этот символ установлен в `true`.
- `LOG_WARNING_ENABLE` - включить вывод в поток предупреждений. По умолчанию этот символ установлен в `true`.
- `LOG_ERROR_ENABLE` - включить вывод в поток ошибок. По умолчанию этот символ установлен в `true`.
- `LOG_FATAL_ENABLE` - включить вывод в поток критических ошибок. По умолчанию этот символ установлен в `true`.

2.2 Конфигурирование времени выполнения

Также, во время выполнения можно установить следующие свойства `Logger`:

- - установить имя файла, в который направляется стандартный вывод. Соотвествующий метод:

  ```
  void Logger::setOutputFile(const char* fname) throw(Logger::IOException)
  ```

  Этот метод генерирует прерывание `IOException` в случае неудачи. `IOException.what()` возвращает в строке сообщение об ошибке.

- - установить дополнительный вывод сообщений на терминал.

  ```
  void Logger::setDuppedToStderr(bool x)
  ```

- - отключить/включить генерирование syslog сообщений.

  ```
  void Logger::setSysLogOutput(bool x)
  ```

Этот метод устанавливает флаг генерирования `SysLog` сообщений. По умолчанию, Logger записывает сообщение в системный журнал (для UNIX). `setSysLogOutput(false)` отключает это свойство, `setSysLogOutput(true)` – включает. Заметим, что в Windows NT эта опция не имеет никакого эффекта.

3 Простейший пример

иллюстрирующий использование пакета Logger приведен ниже:

```
#define LOG_DEBUG_ENABLE true

#include <GradSoft/Logger.h>

void debug_callback(const char* msg)
{
 cerr << "debug_callback:" << msg << endl;
}

int main(int argc, char** argv)
{
 try {
  GradSoft::Logger logger("file.log");
  logger.setCallback(GradSoft::Debug,debug_callback);

  logger.debugs() << "debug output 1 for " << argv[0] << endl;
 }catch(Logger::IOException){
   cerr << "can't open log file" << endl;
   return 1;
 }

 return 0;
}
```

4 Использование Logger в мультипоточном режиме

Вы можете использовать `Logger` в мультипоточных программах: вызов всех методов `Logger` безопасен; При обращении к потокам вывода Logger существует потенциальноая опасность интерференции сообщений, поступающих из разных потоков. Для предотвращения этого мы резервируем для каждого типа сообщений соответствующий mutex и определяем класс - блокировщик доступа к этому mutex-у, который блокирует его при создании и разблокирует при удаленнии.

Соответствующий фрагмент кода приведен ниже:

```
{
  Logger::DebugLocker guard(logger.debugs());
  logger.debugs() << "print " << what << " you want";
}
```

Теперь повторим то-же самое более формально с указанием схемы наименования:
Для каждого типа событий `<Xxx>EventType` определен класс `Loggex::<Xxx>Locker` у которого определенны следующие методы:

- `XxxLocker(XxxStream&)` - конструктор, принимает во владение mutex, блокирующий операторы вывода в соответствующий поток,
- `~XxxLocker()` - освобождает его.

В случае когда вывод в соответствующи поток отключен, этот класс редуцируется до пустого класса с пустыми операциями.

5 Требования к программному окружению

1. Ваша стандартная билиотека C++ должна содержать класс string.
2. Несколько переменных макропроцессора определены в файле `LoggerConfig.h`, генерируемом при компиляции пакета (для UNIX), и `LoggerConfigNT.h` для NT. Потенциально возможен конфликт между определениями в этом файле и определениями из других макропакетов. Для того, чтобы этого не произошло мы рекомендуем заключать ваши макроопределения autoconf в предложения условной компиляции:

   ```
   #ifndef HAVE_Xxx
   #undef HAVE_Xxx
   #endif
   ```

3. Если Вы работаете под управлением Windows NT, Вам необходимо:
   (a) определить макрос WIN32 перед включением файла Logger.h
   (b) использовать только "новые" библиотеки и, соответственно, заголовочные файлы iostream, fstream и т.п. вместо аналогичных iostream.h, fstream.h и т.п.

6 Перечень изменений

03.01.2002 - обновление в соответствии с новой версией GradC++ToolBox 1.4.0

03.07.2001 - изменен пример: заголовочный файл `GradSoft/Logger` больше не используется.

29.05.2001 - изменение структуры, добавлена глава о установках свойств времени выполнения, отражена смена требований к программному окружению.

18.02.2001 - просмотр, добавлены формальные атрибуты эксплуатационной документации.

09.08.2000 - первая версия.