Łukasz Kolber

Projekt C++

Temat Projektu

Trudność Projektu: Średnia
Nazwa: Logi systemowe
Opis: Opracować bibliotekę umożliwiającą zapis logów generowanych w trakcie pracy programu zarówno lokalnie w pliku, jak i przesyłanie ich zdalnie poprzez sieć. Przygotować przykładowe programy korzystające z takiej biblioteki.

Milestone 1

Funkcje

Podstawą tego projektu będzie zapewnienie możliwości rejestrowania sytuacji w systemie na różnych poziomach takich jak:

- debug
- info
- warning
- error
- critical

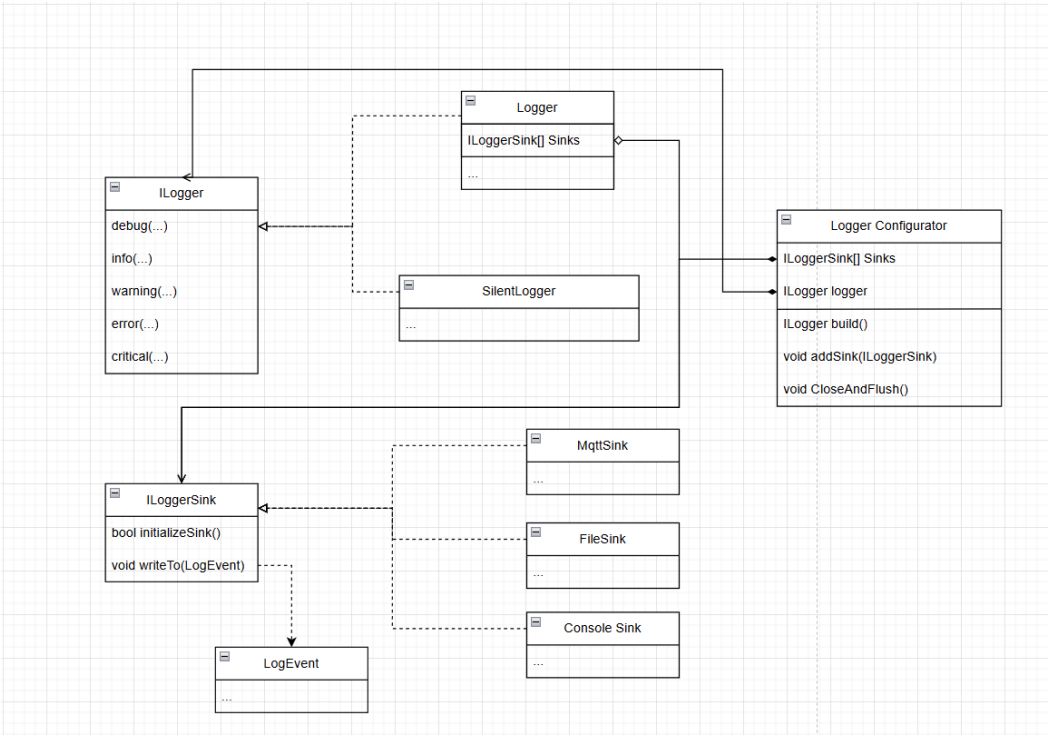
Podstawą do zapisu zdarzeń będzie ich wpisanie do konsoli oraz zapisanie pliku w systemie. Natomiast możliwość zdalnego zapisu poprzez sieć będzie zrealizowana przez co najmniej jedną z opcji:

- baza danych Postgres
- wysłanie wiadomosc na topic MQTT
- wysłanie zapytania REST

Komfortowe dla użytkownika również będzie możliwość zapisywania logów do kilku miejsc na raz.

Jako że w tym projekcie nie będzie elementów wizualnych bo jest to biblioteka do logów przygotowałem wstępny szkic najważniejszych klas. Po krótce opisze teraz za co będa odpowiadały poszczególne elementy:

- ILogger - interfejs klas służących do rejestrowania zdarzeń w systemie (podstawowa implementacja to Logger do obsługi kilku ‘zlewów’)
- ILoggerSink - zlew który będzie odpowiadał za zapisywanie zdarzeń w określonych źródłach lub przesyłanie ich dalej (implementacje to np Console Sink czy File Sink)
- Logger Configurator - klasa odpowiedzialna za tworzenie loggera i przechowywanie aktualnej instancji dostępnej z każdego miejsca (tutaj raczej będzie to rozbite na 2 klasy, ale to się wyprostuje w miare implementacji)
- Log Event - zdarzenie do zalogowania



Rys. 1. Wstępny szkic klas

Przykłady Zastosowania

Realizacja projektu obejmuje również jakiś przykład zastosowania biblioteki do logów więc planuje zrealizować:

- program logujący wartości z danego zakresu - prezentacja działania Console Sink
- program uruchamiający kilka funkcji na kilku wątkach które wpisują do pliku liczby z danego zakresu - prezentacja działania File Sink
- program wysyłający logi co określony czas zdalnie i drugi program który odczytuje / odbiera te logi - prezentacja działania MQTT/REST/Postgres Sink

Środowisko

Najprawdopodobniej Visual Studio 2022 z kompilatorem MSVC, ale możliwe też jest użycie środowiska CLion.

Uwagi do Milestone 1

Ocenione przez	MS Michał Syfert
Komentarz zwrotny	– Może interfejs graficzny nie będzie potrzebny ale jakiś interfejs do testów przyda się. A propozycja opisu struktur danych jaki będzie zastosowany? Proszę jakoś rozbudować program - przedstawiony zakres to trochę za mało. Propozycja klas to na razie za wcześnie - to etap II.

Milestone 2

Rozbudowane Funkcje

Dodatkowe funkcje w stosunku do MS1 zaznaczone kursywą

Podstawowe funkcje projektu:

- rejestrowanie zdarzeń na różnych poziomach (debug, info, warning, error, critical)
- *możliwość wyboru formatu w jakim są logowane informacje (text, json)*
- dostępne zlewy
    - konsola (stdout)
    - plik
    - *plik z rotacja - co określoną zapisaną wielkość powstaje nowy plik z logami i można podac maksymalna liczbe przechowywanych plików historycznych*
    - **serwer http 1.1 - logger jako klient wykonuje request http do określonego serwera - *doprecyzowanie w odniesieniu do Milestone 1 implementacja + test, udokumentowane endpointy dla api***
- możliwość konfiguracji kilku ‘zlewów‘ na logi na raz
- *możliwość zawężenie poziomu logowania w konkretnym zlewie*
- *możliwość dołączania thread id informacji logu*
- *możliwość dołączania nazwy funkcji (linii kodu) w której została wykonana funkcja logująca*
- *interfejs testowy google test - dla testów sprawdzających działanie podstawowych elementów (czy zapisuje do pliku, czy respektuje poziomy logowania)*
- przykład konfiguracji i zastosowania biblioteki
    - pokazanie jak działa podstawowy zlew
    - przykład konfiguracji biblioteki loggera dla biblioteki
    - pokazanie współpracy kilku zlewów na raz
    - pokazanie jak działają zlewy z kilkoma wątkami

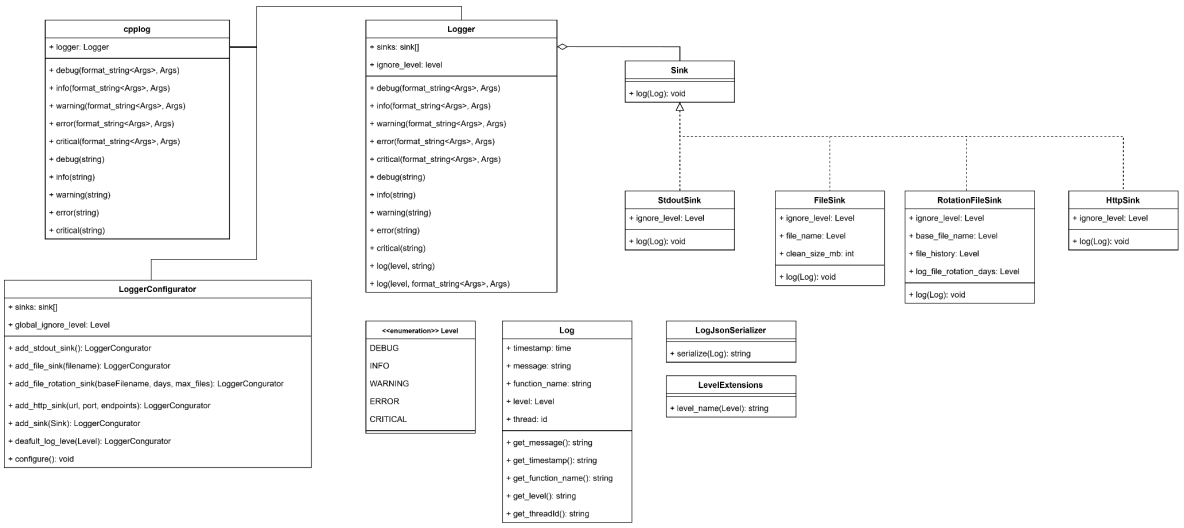
Środowisko

Zastosowany zostanie Clion wraz z MINGW oraz Google Test do testów. Używana też będzie biblioteka standardowa std.

Struktury Danych

Podstawową strukturą danych będzie klasa Log której celem będzie zbieranie i przetrzymywanie informacji o zdarzeniu w systemie, która będzie przekazywana do klas odpowiedzialnych za utrwalenie zdarzenia. Dostępny będzie także enumeration Level określający poziom zdarzenia z perspektywy programisty.

Klasy



Rys. 2. Szkic klas

(Nieuwzględnianiam klas które będą wykorzystane w testach)

Klasy i ich zadania

Logger

Początkowo widziałem klasę logger jako publiczny interfejs niestety nie jest to możliwe przez zastosowanie templatów. Klasa będzie posiadać listę skonfigurowanych ‘zlewów’ na początku działania, podstawowy poziom odrzucania zdarzenia oraz tworzenie zdarzenia i przesłanie go dalej do każdego ‘zlewu’.

Sink i Implementacje

Klasa Sink jest to podstawowy interfejs służący do definiowania zlewów, zdanie jest bardzo proste zagwarantować iż klasy pochodne będą implementowały funkcje przesyłającą zdarzenie, implementacje:

- Stdout sink - zlew którego zadaniem jest wypisanie sformatowanej informacji na ekran
- File Sink - zlew którego podstawowym zadaniem jest zapisanie sformatowanej informacji do pliku, uwzględniając rozmiar pliku przy którym plik zostanie ponownie utworzony
- Rotation File Sink - zlew podobny do File Sink, ale będzie przechowywał historyczne pliki zlewów np. ostatnie 3 przy ograniczeniu określonego rozmiaru pliku
- Http Sink - zlew który bedzie wysyłał zapytanie http na określony endpoint przesyłając dane w formacie json

Logger Configurator

Klasa odpowiedzialna za tworzenie klasy logger, bedzie zbierac podstawowe informacje takie jak skonfigurowane zlewy czy poziom logowania. Finalną metodą będzie metoda configure która stworzy klase logger oraz utrwali ja w postaci statycznego pola.

Cpplog

Klasa zawierająca statyczne metody służące do logowania aby uprościć proces dostępu do funkcji logujących i klasy loggera. Klasa będzie również przechowywać wskazanie na aktualnie defaultową klasę loggera, która bedzie konfigurowana z pomocą logger configuration.

Inne Klasy

- Level - enumeration określający poziom zdarzenia
- Log - klasa zbierająca informacje o zdarzeniu takie jak, poziom, czas, otoczenie - będzie to główna struktura danych przekazywana pomiędzy różnymi komponentami systemu
- LogJsonSerializer - klasa odpowiedzialna za utworzenie jsona bazując na klasie Log
- LevelExtensions - zbiór rozszerzeń dla klasy

Interfejs

Biblioteka nie posiada interfejsu graficznego, jedynie będzie interfejs testów. Myślę że nie ma sensu wypisywać jakieś konkretne scenariusze testowe się pojawią.

Uwagi

Serwer http odbierający zapytanie będzie napisany w c#.
Interfejs Testowy dodany - google test.
Struktury danych (nie do końca wiedziałem o co tu chodzi) - klasa Log i Level.

Dodano nowe funkcjonalności. - W przypadku gdy liczba funkcjonalności dalej jest za mała poprosze o jakieś wskazówki co mogło by się jeszcze przydać.

Uwagi do Milestone 2

Ocenione przez	MS Michał Syfert
Komentarz zwrotny	Program dobrze się rozwija. Tak trzymać.

Milestone 3

Końcowa Idea

Końcowa idea która stała za biblioteką to stworzenie rozwiązania które będzie można prosto użyć z różnym kompilatorem i ewentualnie w miarę potrzeby stworzyć własny zlew realizujący konkretne zadania - z tego powodu zlew Http został przeniesiony z kodu biblioteki do kodu przykładu. Biblioteka pozwala w prosty sposób skonfigurować podstawowy zlew poprzez kod c++, natomiast dostęp do loggera z każdego miejsca zapewnia klasa statyczna Cpplog. Biblioteka umożliwia nadpisanie tego co jest logowane przez klasę opcji i w jakim formacie text/json. Podstawa biblioteki zawiera również kilka gotowych do użycia zlewów z możliwością ich konfiguracji.

Zmiany w stosunku do MS2

- **HttpSink** został wydzielony z biblioteki do przykładu (kod biblioteki nie jest zależny od jednej platformy)
- Dodane klasy opcji które agregują możliwości konfiguracji dla konkretnych zlewów
- Dodanie klasy formatter której zadaniem jest przekształcić informacje do odpowiedniego formatu
- Dodanie Readme zawierającego podstawowe informacje
- zastosowanie cmake do budowy rozwiązania

Podsumowanie

Projekt jest dostępny w repozytorium Github: Cpplog. Można tam znaleźć również zaktualizowany diagram klas.

Wymóg	Jak spełniony
Zastosowanie programowania obiektowego	Tak - widoczne na diagramie klas
Dziedziczenie	Tak - implementacje Sink oraz opcje dla plików
Polimorfizm	Tak - np. klasa logger i cpplog
Podział na .h oraz .cpp	Tak - podział na src oraz include/cpplog.
Udokumentowany interfejs publiczny	Tak - pliki w include/cpplog
Uzycie biblioteki opartej o szablony	Tak - std.
Rozdział Logiki GUI	Nie dotyczy
Własne szablony	Tak w klasie logger i cpplog
Repozytorium Kodu	Tak - git, github.