

Kursach

Создано системой Doxygen 1.9.4

1 Иерархический список классов	1
1.1 Иерархия классов	1
2 Алфавитный указатель классов	3
2.1 Классы	3
3 Список файлов	5
3.1 Файлы	5
4 Классы	7
4.1 Класс ClientHandler	7
4.1.1 Подробное описание	7
4.1.2 Методы	7
4.1.2.1 compute_md5()	7
4.1.2.2 create_salt()	8
4.1.2.3 manage_connection()	8
4.2 Класс ConnectorRefactored	9
4.2.1 Подробное описание	9
4.2.2 Методы	9
4.2.2.1 connect_to_registry()	9
4.2.2.2 get_registry()	9
4.3 Класс crit_err	10
4.3.1 Подробное описание	11
4.3.2 Конструктор(ы)	11
4.3.2.1 crit_err()	11
4.4 Класс InterfaceRefactored	11
4.4.1 Подробное описание	11
4.4.2 Методы	11
4.4.2.1 process_command()	12
4.5 Класс Logger	13
4.5.1 Подробное описание	13
4.5.2 Конструктор(ы)	14
4.5.2.1 Logger()	14
4.5.3 Методы	15
4.5.3.1 get_current_datetime()	15
4.5.3.2 set_path()	15
4.5.3.3 writelog()	15
4.6 Класс MathEngine	16
4.6.1 Подробное описание	16
4.6.2 Конструктор(ы)	16
4.6.2.1 MathEngine()	16
4.6.3 Методы	17
4.6.3.1 retrieve_result()	17
4.7 Класс no_crit_err	17

4.7.1 Подробное описание	18
4.7.2 Конструктор(ы)	18
4.7.2.1 no_crit_err()	18
5 Файлы	19
5.1 Файл CalculatorRefactored.h	19
5.1.1 Подробное описание	19
5.2 CalculatorRefactored.h	20
5.3 ClientHandler.h	20
5.4 Файл ConnectorRefactored.h	21
5.4.1 Подробное описание	21
5.5 ConnectorRefactored.h	22
5.6 Файл Errors.h	22
5.6.1 Подробное описание	23
5.7 Errors.h	23
5.8 Файл InterfaceRefactored.h	24
5.8.1 Подробное описание	24
5.9 InterfaceRefactored.h	25
5.10 Файл LoggerRefactored.h	25
5.10.1 Подробное описание	26
5.11 LoggerRefactored.h	26
Предметный указатель	27

Глава 1

Иерархический список классов

1.1 Иерархия классов

Иерархия классов.

ClientHandler	7
ConnectorRefactored	9
InterfaceRefactored	11
Logger	13
MathEngine	16
std::runtime_error	
crit_err	10
no_crit_err	17

Глава 2

Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

ClientHandler	Класс для управления соединениями клиентов	7
ConnectorRefactored	Класс для подключения к реестру	9
crit_err	Класс для обработки критических ошибок	10
InterfaceRefactored	Интерфейс для обработки командной строки	11
Logger	Класс для записи логов	13
MathEngine	Класс для математических вычислений	16
no_crit_err	Класс для обработки некритических ошибок	17

Глава 3

Список файлов

3.1 Файлы

Полный список документированных файлов.

CalculatorRefactored.h	19
ClientHandler.h	??
ConnectorRefactored.h	21
Errors.h	22
InterfaceRefactored.h	24
LoggerRefactored.h	25

Глава 4

Классы

4.1 Класс ClientHandler

Класс для управления соединениями клиентов.

```
#include <ClientHandler.h>
```

Открытые члены

- `int manage_connection (int port, const char *registry_path, const char *log_path, Logger *logger)`
Управление соединениями на заданном порту.

Открытые статические члены

- `static std::string create_salt ()`
Создать соль для хеширования.
- `static std::string compute_md5 (const std::string &input)`
Вычислить MD5 хеш от строки.

4.1.1 Подробное описание

Класс для управления соединениями клиентов.

4.1.2 Методы

4.1.2.1 `compute_md5()`

```
std::string ClientHandler::compute_md5 (  
    const std::string & input ) [static]
```

Вычислить MD5 хеш от строки.

Аргументы

input	Строка для хеширования.
-------	-------------------------

Возвращает

Хешированная строка.

4.1.2.2 create_salt()

```
std::string ClientHandler::create_salt ( ) [static]
```

Создать соль для хеширования.

Возвращает

Сгенерированная строка соли.

4.1.2.3 manage_connection()

```
int ClientHandler::manage_connection (
    int port,
    const char * registry_path,
    const char * log_path,
    Logger * logger )
```

Управление соединениями на заданном порту.

Аргументы

port	Порт сервера для прослушивания.
registry_path	Путь к файлу реестра клиентов.
log_path	Путь к файлу логов.
logger	Указатель на объект логгера.

Возвращает

Код результата операции.

Объявления и описания членов классов находятся в файлах:

- ClientHandler.h
- ClientHandler.cpp

4.2 Класс ConnectorRefactored

Класс для подключения к реестру.

```
#include <ConnectorRefactored.h>
```

Открытые члены

- `int connect_to_registry (std::string path="/home/stud/kursach2/base/base.txt")`
Подключиться к реестру по заданному пути.
- `std::map< std::string, std::string > get_registry ()`
Получить реестр пользователей.

Закрытые данные

- `std::map< std::string, std::string > registry`
Реестр пользователей.

4.2.1 Подробное описание

Класс для подключения к реестру.

4.2.2 Методы

4.2.2.1 connect_to_registry()

```
int ConnectorRefactored::connect_to_registry (
    std::string path = "/home/stud/kursach2/base/base.txt" )
```

Подключиться к реестру по заданному пути.

Аргументы

path	Путь к файлу реестра.
------	-----------------------

Возвращает

Код результата операции.

4.2.2.2 get_registry()

```
std::map< std::string, std::string > ConnectorRefactored::get_registry ( )
```

Получить реестр пользователей.

Возвращает

Реестр в виде карты.

Объявления и описания членов классов находятся в файлах:

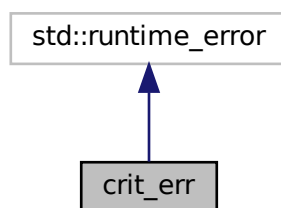
- [ConnectorRefactored.h](#)
- [ConnectorRefactored.cpp](#)

4.3 Класс `crit_err`

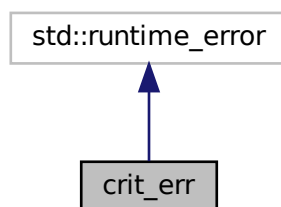
Класс для обработки критических ошибок.

```
#include <Errors.h>
```

Граф наследования: `crit_err`:



Граф связей класса `crit_err`:



Открытые члены

- [crit_err](#) (`const std::string &message`)
Конструктор критической ошибки.

4.3.1 Подробное описание

Класс для обработки критических ошибок.

4.3.2 Конструктор(ы)

4.3.2.1 crit_err()

```
crit_err::crit_err (
    const std::string & message ) [explicit]
```

Конструктор критической ошибки.

Аргументы

message	Сообщение об ошибке.
---------	----------------------

Объявления и описания членов класса находятся в файле:

- [Errors.h](#)

4.4 Класс InterfaceRefactored

Интерфейс для обработки командной строки.

```
#include <InterfaceRefactored.h>
```

Открытые члены

- InterfaceRefactored ()
Конструктор по умолчанию.
- int [process_command](#) (int argc, const char **argv, bool is_test=false)
Обработка командной строки.

4.4.1 Подробное описание

Интерфейс для обработки командной строки.

4.4.2 Методы

4.4.2.1 process_command()

```
int InterfaceRefactored::process_command (
    int argc,
    const char ** argv,
    bool is_test = false )
```

Обработка командной строки.

Аргументы

argc	Количество аргументов.
argv	Аргументы командной строки.
is_test	Режим тестирования.

Возвращает

Код успешного выполнения.

Объявления и описания членов классов находятся в файлах:

- [InterfaceRefactored.h](#)
- InterfaceRefactored.cpp

4.5 Класс Logger

Класс для записи логов.

```
#include <LoggerRefactored.h>
```

Открытые члены

- int [writelog](#) (std::string message)
Записать сообщение в лог.
- int [set_path](#) (std::string file_path)
Установить путь к файлу логов.
- [Logger](#) ()
Конструктор по умолчанию.
- [Logger](#) (std::string path)
Конструктор с указанием пути.

Закрытые статические члены

- static std::string [get_current_datetime](#) (std::string format)
Получить текущую дату и время.

Закрытые данные

- std::string log_file_path
Путь к файлу логов.

4.5.1 Подробное описание

Класс для записи логов.

4.5.2 Конструктор(ы)

4.5.2.1 Logger()

```
Logger::Logger (  
    std::string path )
```

Конструктор с указанием пути.

Аргументы

path	Путь к файлу логов.
------	---------------------

4.5.3 Методы

4.5.3.1 get_current_datetime()

```
string Logger::get_current_datetime (
    std::string format ) [static], [private]
```

Получить текущую дату и время.

Аргументы

format	Формат вывода.
--------	----------------

Возвращает

Строка с текущей датой и временем.

4.5.3.2 set_path()

```
int Logger::set_path (
    std::string file_path )
```

Установить путь к файлу логов.

Аргументы

file_path	Новый путь к файлу логов.
-----------	---------------------------

Возвращает

Код успешного выполнения.

4.5.3.3 writelog()

```
int Logger::writelog (
    std::string message )
```

Записать сообщение в лог.

Аргументы

message	Сообщение для записи.
---------	-----------------------

Возвращает

Код успешной записи.

Объявления и описания членов классов находятся в файлах:

- [LoggerRefactored.h](#)
- [LoggerRefactored.cpp](#)

4.6 Класс MathEngine

Класс для математических вычислений.

```
#include <CalculatorRefactored.h>
```

Открытые члены

- [MathEngine](#) (std::vector< int64_t > data)
Конструктор, вычисляющий сумму квадратов элементов.
- int64_t [retrieve_result](#) ()
Получить вычисленное значение.

Закрытые данные

- int64_t computed_value
Значение, вычисляемое с помощью методов класса.

4.6.1 Подробное описание

Класс для математических вычислений.

4.6.2 Конструктор(ы)

4.6.2.1 MathEngine()

```
MathEngine::MathEngine (  
    std::vector< int64_t > data )
```

Конструктор, вычисляющий сумму квадратов элементов.

Аргументы

data	Вектор целых чисел для обработки.
------	-----------------------------------

4.6.3 Методы

4.6.3.1 retrieve_result()

```
int64_t MathEngine::retrieve_result ( )
```

Получить вычисленное значение.

Возвращает

Вычисленное значение.

Объявления и описания членов классов находятся в файлах:

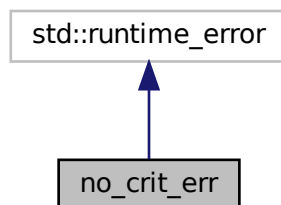
- [CalculatorRefactored.h](#)
- CalculatorRefactored.cpp

4.7 Класс no_crit_err

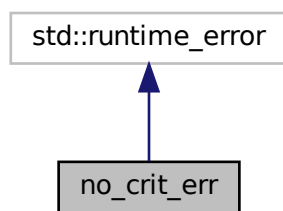
Класс для обработки некритических ошибок.

```
#include <Errors.h>
```

Граф наследования: no_crit_err:



Граф связей класса `no_crit_err`:



Открытые члены

- `no_crit_err` (`const std::string &message`)
Конструктор не критической ошибки.

4.7.1 Подробное описание

Класс для обработки не критических ошибок.

4.7.2 Конструктор(ы)

4.7.2.1 `no_crit_err()`

```
no_crit_err::no_crit_err (  
    const std::string & message ) [explicit]
```

Конструктор не критической ошибки.

Аргументы

<code>message</code>	Сообщение об ошибке.
----------------------	----------------------

Объявления и описания членов класса находятся в файле:

- [Errors.h](#)

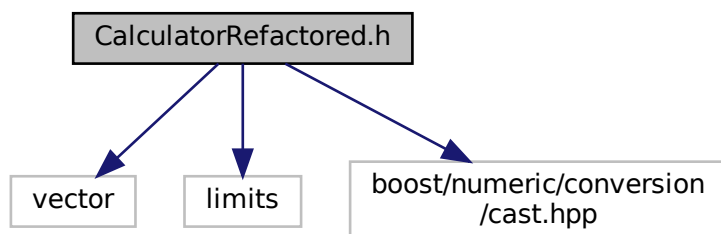
Глава 5

Файлы

5.1 Файл CalculatorRefactored.h

```
#include <vector>
#include <limits>
#include <boost/numeric/conversion/cast.hpp>
```

Граф включаемых заголовочных файлов для CalculatorRefactored.h:



Классы

- class [MathEngine](#)

Класс для математических вычислений.

5.1.1 Подробное описание

Автор

Сверчков А.Д.

Версия

1.0

Дата

26.11.2024

Авторство

ИБСТ ПГУ

Предупреждения

Это учебный пример

5.2 CalculatorRefactored.h

[См. документацию.](#)

```
1
8 #pragma once
9 #include <vector>
10 #include <limits>
11 #include <boost/numeric/conversion/cast.hpp>
12
16 class MathEngine
17 {
19     int64_t computed_value;
20 public:
21
27     MathEngine(std::vector<int64_t> data);
28
34     int64_t retrieve_result();
35 };
```

5.3 ClientHandler.h

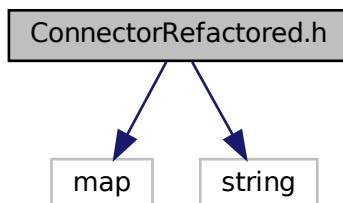
```
1 #pragma once
2 #include <string>
3 #include <map>
4 #include "LoggerRefactored.h"
5 #include "Errors.h"
6
10 class ClientHandler
11 {
12 public:
13
24     int manage_connection(int port, const char* registry_path, const char* log_path, Logger* logger);
25
31     static std::string create_salt();
32
40     static std::string compute_md5(const std::string& input);
41 };
```


5.4 Файл ConnectorRefactored.h

```
#include <map>
```

```
#include <string>
```

Граф включаемых заголовочных файлов для ConnectorRefactored.h:



Классы

- class [ConnectorRefactored](#)

Класс для подключения к реестру.

5.4.1 Подробное описание

Автор

Сверчков А.Д.

Версия

1.0

Дата

26.11.2024

Авторство

ИБСТ ПГУ

Предупреждения

Это учебный пример

5.5 ConnectorRefactored.h

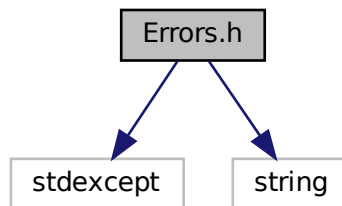
См. документацию.

```
1
8 #pragma once
9 #include <map>
10 #include <string>
11
15 class ConnectorRefactored
16 {
17 private:
19     std::map<std::string, std::string> registry;
20 public:
21
29     int connect_to_registry(std::string path = "/home/stud/kursach2/base/base.txt");
30
36     std::map<std::string, std::string> get_registry();
37 };
```

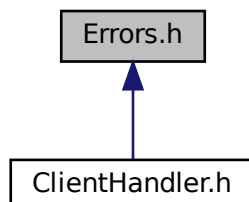
5.6 Файл Errors.h

```
#include <stdexcept>
#include <string>
```

Граф включаемых заголовочных файлов для Errors.h:



Граф файлов, в которые включается этот файл:



Классы

- class `crit_err`
Класс для обработки критических ошибок.
- class `no_crit_err`
Класс для обработки некритических ошибок.

5.6.1 Подробное описание

Автор

Сверчков А.Д.

Версия

1.0

Дата

26.11.2024

Авторство

ИБСТ ПГУ

Предупреждения

Это учебный пример

5.7 Errors.h

[См. документацию.](#)

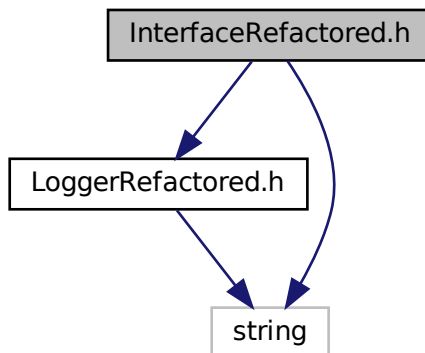
```
1
8 #pragma once
9 #include <stdexcept>
10 #include <string>
11
15 class crit_err : public std::runtime_error {
16 public:
22     explicit crit_err(const std::string& message);
23 };
24
28 class no_crit_err : public std::runtime_error {
29 public:
35     explicit no_crit_err(const std::string& message);
36 };
```

5.8 Файл InterfaceRefactored.h

```
#include "LoggerRefactored.h"
```

```
#include <string>
```

Граф включаемых заголовочных файлов для InterfaceRefactored.h:



Классы

- class [InterfaceRefactored](#)
Интерфейс для обработки командной строки.

5.8.1 Подробное описание

Автор

Сверчков А.Д.

Версия

1.0

Дата

26.11.2024

Авторство

ИБСТ ПГУ

Предупреждения

Это учебный пример

5.9 InterfaceRefactored.h

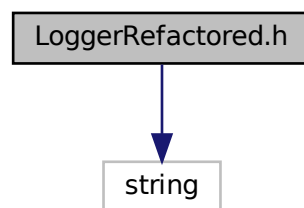
См. документацию.

```
1
8 #pragma once
9 #include "LoggerRefactored.h"
10 #include <string>
11
15 class InterfaceRefactored {
16 public:
18     InterfaceRefactored();
19
29     int process_command(int argc, const char** argv, bool is_test = false);
30 };
```

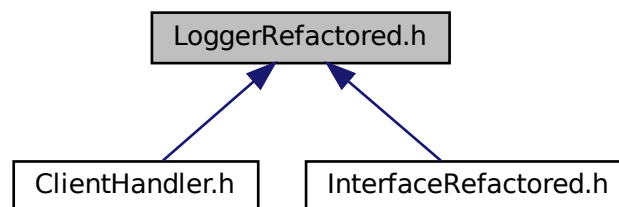
5.10 Файл LoggerRefactored.h

```
#include <string>
```

Граф включаемых заголовочных файлов для LoggerRefactored.h:



Граф файлов, в которые включается этот файл:



Классы

- class [Logger](#)

Класс для записи логов.

5.10.1 Подробное описание

Автор

Сверчков А.Д.

Версия

1.0

Дата

26.11.2024

Авторство

ИБСТ ПГУ

Предупреждения

Это учебный пример

5.11 LoggerRefactored.h

[См. документацию.](#)

```
1
8 #pragma once
9 #include <string>
10
14 class Logger
15 {
23     static std::string get_current_datetime(std::string format);
24
26     std::string log_file_path;
27
28 public:
29
37     int writelog(std::string message);
38
46     int set_path(std::string file_path);
47
49     Logger();
50
56     Logger(std::string path);
57 };
```

Предметный указатель

- CalculatorRefactored.h, [19](#)
- ClientHandler, [7](#)
 - compute_md5, [7](#)
 - create_salt, [8](#)
 - manage_connection, [8](#)
- compute_md5
 - ClientHandler, [7](#)
- connect_to_registry
 - ConnectorRefactored, [9](#)
- ConnectorRefactored, [9](#)
 - connect_to_registry, [9](#)
 - get_registry, [9](#)
- ConnectorRefactored.h, [21](#)
- create_salt
 - ClientHandler, [8](#)
- crit_err, [10](#)
 - crit_err, [11](#)
- Errors.h, [22](#)
- get_current_datetime
 - Logger, [15](#)
- get_registry
 - ConnectorRefactored, [9](#)
- InterfaceRefactored, [11](#)
 - process_command, [11](#)
- InterfaceRefactored.h, [24](#)
- Logger, [13](#)
 - get_current_datetime, [15](#)
 - Logger, [14](#)
 - set_path, [15](#)
 - writelog, [15](#)
- LoggerRefactored.h, [25](#)
- manage_connection
 - ClientHandler, [8](#)
- MathEngine, [16](#)
 - MathEngine, [16](#)
 - retrieve_result, [17](#)
- no_crit_err, [17](#)
 - no_crit_err, [18](#)
- process_command
 - InterfaceRefactored, [11](#)
- retrieve_result
 - MathEngine, [17](#)
- set_path
 - Logger, [15](#)
- writelog
 - Logger, [15](#)