

Project Documentation

Henrik Sørensen

November 18, 2023

Contents

1	EnvVar Class	2
1.1	Source Code	2
2	ConfigManager Class	3
2.1	Source Code	3
3	Logger Class	7
3.1	Source Code	7
4	testair	11
4.1	Source Code	11
4.1.1	testair.hpp	11
4.1.2	testair.cpp	12

1 EnvVar Class

1.1 Source Code

```
1  /*
2   This file is part of the AppFramework project.
3
4   AppFramework is free software: you can redistribute it and/or modify
5   it under the terms of the GNU General Public License as published by
6   the Free Software Foundation, GPL version 4.
7
8   AppFramework is distributed in the hope that it will be useful,
9   but WITHOUT ANY WARRANTY; without even the implied warranty of
10  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11  GNU General Public License version 4 for more details.
12
13  You should have received a copy of the GNU General Public License
14  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
15  */
16
17  // EnvVar.hpp
18
19  #ifndef ENVVAR_HPP
20  #define ENVVAR_HPP
21
22  #include <string>
23  #include <optional>
24  #ifdef THREAD_SAFE
25  #include <mutex>
26  #endif
27
28  class EnvVar {
29  public:
30      explicit EnvVar(const std::string& name);
31
32      std::string get() const;
33      bool set(const std::string& value) const;
34      void store();
35      bool restore() const;
36
37  private:
38      std::string varName;
39      std::optional<std::string> storedValue;
40  #ifdef THREAD_SAFE
41      static std::mutex mtx; // Mutex for thread safety
42  #endif
43  };
44
45  #endif // ENVVAR_HPP

```



```
1  /*
2   This file is part of the AppFramework project.
3
4   AppFramework is free software: you can redistribute it and/or modify
5   it under the terms of the GNU General Public License as published by
6   the Free Software Foundation, GPL version 4.
7
```

```

8   AppFramework is distributed in the hope that it will be useful,
9   but WITHOUT ANY WARRANTY; without even the implied warranty of
10  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11  GNU General Public License version 4 for more details.
12
13  You should have received a copy of the GNU General Public License
14  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
15  */
16
17  // EnvVar.cpp
18
19  #include "EnvVar.hpp"
20  #include "Logger.hpp"
21  #include <cstdlib>
22  #ifndef THREAD_SAFE
23  #include <mutex>
24  std::mutex EnvVar::mtx; // Define the static mutex
25  #endif
26
27  EnvVar::EnvVar(const std::string& name) : varName(name) {}
28
29  std::string EnvVar::get() const {
30  #ifndef THREAD_SAFE
31      std::lock_guard<std::mutex> lock(mtx); // Lock the mutex
32  #endif
33      const char* value = std::getenv(varName.c_str());
34      return (value != nullptr) ? std::string(value) : std::string();
35  }
36
37  bool EnvVar::set(const std::string& value) const {
38  #ifndef THREAD_SAFE
39      std::lock_guard<std::mutex> lock(mtx); // Lock the mutex
40  #endif
41      return setenv(varName.c_str(), value.c_str(), 1) == 0;
42  }
43
44  void EnvVar::store() {
45  #ifndef THREAD_SAFE
46      std::lock_guard<std::mutex> lock(mtx); // Lock the mutex
47  #endif
48      storedValue = get();
49  }
50
51  bool EnvVar::restore() const {
52  #ifndef THREAD_SAFE
53      std::lock_guard<std::mutex> lock(mtx); // Lock the mutex
54  #endif
55      if (storedValue.has_value()) {
56          return set(storedValue.value());
57      }
58      return false;
59  }

```

2 ConfigManager Class

2.1 Source Code

```

1  /*
2   This file is part of the AppFramework project.
3
4   AppFramework is free software: you can redistribute it and/or modify
5   it under the terms of the GNU General Public License as published by
6   the Free Software Foundation, GPL version 4.
7
8   AppFramework is distributed in the hope that it will be useful,
9   but WITHOUT ANY WARRANTY; without even the implied warranty of
10  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11  GNU General Public License version 4 for more details.
12
13  You should have received a copy of the GNU General Public License
14  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
15  */
16
17  // ConfigManager.hpp
18
19  #ifndef CONFIGMANAGER_HPP
20  #define CONFIGMANAGER_HPP
21
22  #include <iostream>
23  #include <string>
24  #include <mutex>
25  #include <nlohmann/json.hpp>
26
27  class ConfigManager {
28  public:
29      explicit ConfigManager(const std::string& configFilePath);
30      ~ConfigManager();
31
32      template<typename T>
33      T get(const std::string& key) const;
34
35      template<typename T>
36      void set(const std::string& key, const T& value);
37
38      void sync();
39
40  #ifdef THREAD_SAFE
41      static std::mutex mtx; // Mutex for thread safety
42  #endif
43
44  private:
45      nlohmann::json config;
46      std::string filePath;
47      const nlohmann::json& getRefToValue(const std::string& key, bool
48          forRead) const;
49      nlohmann::json& getRefToValue(const std::string& key);
49  };
50
51  #endif // CONFIGMANAGER_HPP

```

```

1  /*
2   This file is part of the AppFramework project.
3
4   AppFramework is free software: you can redistribute it and/or modify
5   it under the terms of the GNU General Public License as published by

```

```

6     the Free Software Foundation, GPL version 4.
7
8     AppFramework is distributed in the hope that it will be useful,
9     but WITHOUT ANY WARRANTY; without even the implied warranty of
10    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11    GNU General Public License version 4 for more details.
12
13    You should have received a copy of the GNU General Public License
14    along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
15 */
16
17 // ConfigManager.cpp
18
19 #include "ConfigManager.hpp"
20 #include "Logger.hpp"
21 #include <iostream>
22 #include <fstream>
23 #include <sstream>
24
25 #ifndef THREAD_SAFE
26     std::mutex ConfigManager::mtx;
27 #endif
28
29 ConfigManager::ConfigManager(const std::string& configFilePath) :
30     filePath(configFilePath) {
31     std::ifstream file(filePath);
32     if (file) {
33         try {
34             file >> config;
35         } catch (const nlohmann::json::parse_error& e) {
36             Logger::getInstance().log("JSON parsing error: " + std::
37                 string(e.what()), "ConfigManager::ConfigManager", Logger
38                 ::Severity::Error);
39             std::cerr << "Configuration loading error. Check log file for
40                 details." << std::endl;
41             config = nlohmann::json::object(); // Ensure config is a
42                 valid JSON object
43         }
44     } else {
45         Logger::getInstance().log("Config file not found: " + filePath, "
46             ConfigManager::ConfigManager", Logger::Severity::Warning);
47         std::cerr << "Configuration file missing. A new one will be
48             created." << std::endl;
49         config = nlohmann::json::object(); // Initialize config as an
50             empty object
51     }
52
53     // Additional check to ensure config is not null
54     if (config.is_null()) {
55         config = nlohmann::json::object();
56     }
57 }
58
59 ConfigManager::~ConfigManager() {
60     sync();
61 }

```

```

55 template<typename T>
56 T ConfigManager::get(const std::string& key) const {
57 #ifdef THREAD_SAFE
58     std::lock_guard<std::mutex> lock(mtx);
59 #endif
60     try {
61         const nlohmann::json& ref = getRefToValue(key, true);
62         return ref.get<T>();
63     } catch (const nlohmann::json::out_of_range& e) {
64         // Handle the case where the key does not exist
65         Logger::getInstance().log("Key not found in configuration: " +
66             key, "ConfigManager::get", Logger::Severity::Warning);
67         throw std::runtime_error("Configuration key not found: " + key);
68     } catch (const nlohmann::json::exception& e) {
69         // Handle other JSON exceptions
70         Logger::getInstance().log("Error accessing key '" + key + "': " +
71             e.what(), "ConfigManager::get", Logger::Severity::Error);
72         throw;
73     }
74 }
75
76 template<typename T>
77 void ConfigManager::set(const std::string& key, const T& value) {
78 #ifdef THREAD_SAFE
79     std::lock_guard<std::mutex> lock(mtx);
80 #endif
81     nlohmann::json& ref = getRefToValue(key); // Use non-const ref
82     ref = value;
83 }
84
85 void ConfigManager::sync() {
86 #ifdef THREAD_SAFE
87     std::lock_guard<std::mutex> lock(mtx);
88 #endif
89     std::ofstream file(filePath);
90     if (file) {
91         file << config.dump(4); // Save the JSON in a pretty format
92     }
93 }
94
95 const nlohmann::json& ConfigManager::getRefToValue(const std::string& key
96     , bool forRead) const {
97     const nlohmann::json* j = &config;
98     std::istringstream iss(key);
99     std::string token;
100     while (std::getline(iss, token, '.')) {
101         j = &((*j).at(token));
102     }
103     return *j;
104 }
105
106 nlohmann::json& ConfigManager::getRefToValue(const std::string& key) {
107     nlohmann::json* j = &config;
108     std::istringstream iss(key);
109     std::string token;
110     while (std::getline(iss, token, '.')) {
111         j = &((*j)[token]);
112     }

```

```

109     }
110     return *j;
111 }
112
113 // Explicit template instantiation
114 template int ConfigManager::get<int>(const std::string& key) const;
115 template std::string ConfigManager::get<std::string>(const std::string&
    key) const;
116 template void ConfigManager::set<int>(const std::string& key, const int&
    value);
117 template void ConfigManager::set<std::string>(const std::string& key,
    const std::string& value);

```

3 Logger Class

3.1 Source Code

```

1  /*
2   This file is part of the AppFramework project.
3
4   AppFramework is free software: you can redistribute it and/or modify
5   it under the terms of the GNU General Public License as published by
6   the Free Software Foundation, GPL version 4.
7
8   AppFramework is distributed in the hope that it will be useful,
9   but WITHOUT ANY WARRANTY; without even the implied warranty of
10  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11  GNU General Public License version 4 for more details.
12
13  You should have received a copy of the GNU General Public License
14  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
15  */
16
17  // Logger.hpp
18
19  #ifndef LOGGER_HPP
20  #define LOGGER_HPP
21
22  #include <string>
23  #include <fstream>
24  #include <mutex>
25
26  class Logger {
27  public:
28      enum class Severity {
29          Trace,
30          Debug,
31          Info,
32          Warning,
33          Error,
34          Fatal
35      };
36
37      static Logger& getInstance();
38      void log(const std::string& message, const std::string& location,
          Severity severity);

```

```

39
40 private:
41     std::ofstream logFile;
42     std::mutex mtx;
43
44     Logger(); // Private constructor for Singleton pattern
45     ~Logger();
46     Logger(const Logger&) = delete;
47     Logger& operator=(const Logger&) = delete;
48
49     std::string severityToString(Severity severity);
50 };
51
52 #endif // LOGGER_HPP

```



```

1  /*
2   This file is part of the AppFramework project.
3
4   AppFramework is free software: you can redistribute it and/or modify
5   it under the terms of the GNU General Public License as published by
6   the Free Software Foundation, GPL version 4.
7
8   AppFramework is distributed in the hope that it will be useful,
9   but WITHOUT ANY WARRANTY; without even the implied warranty of
10  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11  GNU General Public License version 4 for more details.
12
13  You should have received a copy of the GNU General Public License
14  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
15  */
16
17  // Logger.cpp
18
19  #include "Logger.hpp"
20  #include "EnvVar.hpp"
21  #include <iostream>
22  #include <fstream>
23  #include <sstream>
24  #include <chrono>
25  #include <iomanip>
26
27  Logger::Logger() {
28      std::string logPath = "testing.log"; // Default log file name
29
30      // Use std::getenv directly to avoid dependency on EnvVar
31      const char* configPath = std::getenv("LOGPATH");
32      if (configPath != nullptr) {
33          logPath = std::string(configPath) + "/testing.log"; // Use the
34                      directory from LOGPATH
35      }
36
37      logFile.open(logPath, std::ios::out | std::ios::app);
38  }
39
40  Logger::~Logger() {
41      if (logFile.is_open()) {
42          logFile.close();
43      }
44  }
45
46  Logger& Logger::getInstance() {
47      static Logger instance;
48      return instance;
49  }
50
51  void Logger::log(const std::string& message, const std::string& location,
52                  Severity severity) {
53      std::lock_guard<std::mutex> lock(mtx);
54
55      // Get current time
56      auto now = std::chrono::system_clock::now();
57      auto now_time_t = std::chrono::system_clock::to_time_t(now);

```

```

56     auto now_localtime = *std::localtime(&now_time_t);
57
58     if (logFile.is_open()) {
59         logFile << "[" << std::put_time(&now_localtime, "%Y-%m-%d %H:%M:%
60             S") << "]" << "
61             << "[" << severityToString(severity) << "]" << "
62             << location << ": " << message << std::endl;
63     }
64
65     std::string Logger::severityToString(Severity severity) {
66         switch (severity) {
67             case Severity::Trace: return "TRACE";
68             case Severity::Debug: return "DEBUG";
69             case Severity::Info: return "INFO";
70             case Severity::Warning: return "WARNING";
71             case Severity::Error: return "ERROR";
72             case Severity::Fatal: return "FATAL";
73             default:
74                 return "UNKNOWN";
75         }
76     }

```

4 testair

4.1 Source Code

4.1.1 testair.hpp

```
1 we use a skel.hpp in the form:
2
3 /*
4  This file is part of the AppFramework project.
5
6  AppFramework is free software: you can redistribute it and/or modify
7  it under the terms of the GNU General Public License as published by
8  the Free Software Foundation, GPL version 4.
9
10 AppFramework is distributed in the hope that it will be useful,
11 but WITHOUT ANY WARRANTY; without even the implied warranty of
12 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 GNU General Public License version 4 for more details.
14
15 You should have received a copy of the GNU General Public License
16 along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
17 */
18
19 // testair.hpp
20
21 #ifndef TESTAIR_HPP
22 #define TESTAIR_HPP
23
24 #ifdef THREAD_SAFE
25 #include <mutex>
26 #endif
27
28 class testair {
29
30 };
31 #endif // TESTAIR_HPP
```

4.1.2 testair.cpp

```
1  /*
2   This file is part of the AppFramework project.
3
4   AppFramework is free software: you can redistribute it and/or modify
5   it under the terms of the GNU General Public License as published by
6   the Free Software Foundation, GPL version 4.
7
8   AppFramework is distributed in the hope that it will be useful,
9   but WITHOUT ANY WARRANTY; without even the implied warranty of
10  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11  GNU General Public License version 4 for more details.
12
13  You should have received a copy of the GNU General Public License
14  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
15  */
16
17  // testair.cpp
18
19  #include "testair.hpp"
20
21  #ifdef THREAD_SAFE
22  #include <mutex>
23  std::mutex EnvVar::mtx; // Define the static mutex
24  #endif
```