Project Documentation

Henrik Sørensen

November 18, 2023

Contents

1 EnvVar Class

1.1 Source Code

```cpp
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// EnvVar.hpp

#ifndef ENVVAR_HPP
#define ENVVAR_HPP

#include <string>
#include <optional>
#ifdef THREAD_SAFE
#include <mutex>
#endif

class EnvVar {
public:
    explicit EnvVar(const std::string& name);

    std::string get() const;
    bool set(const std::string& value) const;
    void store();
    bool restore() const;

private:
    std::string varName;
    std::optional<std::string> storedValue;
#ifdef THREAD_SAFE
    static std::mutex mtx;  // Mutex for thread safety
#endif
};

#endif // ENVVAR_HPP
```

```cpp
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// EnvVar.cpp

#include "EnvVar.hpp"
#include "Logger.hpp"
#include <cstdlib>
#ifdef THREAD_SAFE
#include <mutex>
std::mutex EnvVar::mtx;  // Define the static mutex
#endif

EnvVar::EnvVar(const std::string& name) : varName(name) {}

std::string EnvVar::get() const {
#ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);  // Lock the mutex
#endif
    const char* value = std::getenv(varName.c_str());
    return (value != nullptr) ? std::string(value) : std::string();
}

bool EnvVar::set(const std::string& value) const {
#ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);  // Lock the mutex
#endif
    return setenv(varName.c_str(), value.c_str(), 1) == 0;
}

void EnvVar::store() {
#ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);  // Lock the mutex
#endif
    storedValue = get();
}

bool EnvVar::restore() const {
#ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);  // Lock the mutex
#endif
    if (storedValue.has_value()) {
        return set(storedValue.value());
    }
    return false;
}
```

2 ConfigManager Class

2.1 Source Code

```cpp
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// ConfigManager.hpp

#ifndef CONFIGMANAGER_HPP
#define CONFIGMANAGER_HPP

#include <iostream>
#include <string>
#include <mutex>
#include <nlohmann/json.hpp>

class ConfigManager {
public:
    explicit ConfigManager(const std::string& configFilePath);
    ~ConfigManager();

    template<typename T>
    T get(const std::string& key) const;

    template<typename T>
    void set(const std::string& key, const T& value);

    void sync();

#ifdef THREAD_SAFE
    static std::mutex mtx;  // Mutex for thread safety
#endif

private:
    nlohmann::json config;
    std::string filePath;
    const nlohmann::json& getRefToValue(const std::string& key, bool
        forRead) const;
    nlohmann::json& getRefToValue(const std::string& key);
};

#endif // CONFIGMANAGER_HPP
```

```cpp
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// ConfigManager.cpp

#include "ConfigManager.hpp"
#include "Logger.hpp"
#include <iostream>
#include <fstream>
#include <sstream>

#ifdef THREAD_SAFE
std::mutex ConfigManager::mtx;
#endif

ConfigManager::ConfigManager(const std::string& configFilePath) :
    filePath(configFilePath) {
    std::ifstream file(filePath);
    if (file) {
        try {
            file >> config;
        } catch (const nlohmann::json::parse_error& e) {
            Logger::getInstance().log("JSON parsing error: " + std::
                string(e.what()), "ConfigManager::ConfigManager", Logger
                ::Severity::Error);
            std::cerr << "Configuration loading error. Check log file for
                 details." << std::endl;
            config = nlohmann::json::object(); // Ensure config is a
                valid JSON object
        }
    } else {
        Logger::getInstance().log("Config file not found: " + filePath, "
            ConfigManager::ConfigManager", Logger::Severity::Warning);
        std::cerr << "Configuration file missing. A new one will be
            created." << std::endl;
        config = nlohmann::json::object(); // Initialize config as an
            empty object
    }

    // Additional check to ensure config is not null
    if (config.is_null()) {
        config = nlohmann::json::object();
    }
}

ConfigManager::~ConfigManager() {
    sync();
}

```

```cpp
template<typename T>
T ConfigManager::get(const std::string& key) const {
#ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);
#endif
    try {
        const nlohmann::json& ref = getRefToValue(key, true);
        return ref.get<T>();
    } catch (const nlohmann::json::out_of_range& e) {
        // Handle the case where the key does not exist
        Logger::getInstance().log("Key not found in configuration: " +
            key, "ConfigManager::get", Logger::Severity::Warning);
        throw std::runtime_error("Configuration key not found: " + key);
    } catch (const nlohmann::json::exception& e) {
        // Handle other JSON exceptions
        Logger::getInstance().log("Error accessing key '" + key + "': " +
             e.what(), "ConfigManager::get", Logger::Severity::Error);
        throw;
    }
}

template<typename T>
void ConfigManager::set(const std::string& key, const T& value) {
#ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);
#endif
    nlohmann::json& ref = getRefToValue(key); // Use non-const ref
    ref = value;
}

void ConfigManager::sync() {
#ifdef THREAD_SAFE
    std::lock_guard<std::mutex> lock(mtx);
#endif
    std::ofstream file(filePath);
    if (file) {
        file << config.dump(4);  // Save the JSON in a pretty format
    }
}

const nlohmann::json& ConfigManager::getRefToValue(const std::string& key
    , bool forRead) const {
    const nlohmann::json* j = &config;
    std::istringstream iss(key);
    std::string token;
    while (std::getline(iss, token, '.')) {
        j = &((*j).at(token));
    }
    return *j;
}

nlohmann::json& ConfigManager::getRefToValue(const std::string& key) {
    nlohmann::json* j = &config;
    std::istringstream iss(key);
    std::string token;
    while (std::getline(iss, token, '.')) {
        j = &((*j)[token]);
    }
    return *j;
}

// Explicit template instantiation
template int ConfigManager::get<int>(const std::string& key) const;
template std::string ConfigManager::get<std::string>(const std::string&
    key) const;
template void ConfigManager::set<int>(const std::string& key, const int&
    value);
template void ConfigManager::set<std::string>(const std::string& key,
    const std::string& value);
```

3 Logger Class

3.1 Source Code

```
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// Logger.hpp

#ifndef LOGGER_HPP
#define LOGGER_HPP

#include <string>
#include <fstream>
#include <mutex>

class Logger {
public:
    enum class Severity {
      Trace,
      Debug,
        Info,
        Warning,
        Error,
        Fatal
    };

    static Logger& getInstance();
    void log(const std::string& message, const std::string& location,
        Severity severity);

private:
    std::ofstream logFile;
    std::mutex mtx;

    Logger(); // Private constructor for Singleton pattern
    ~Logger();
    Logger(const Logger&) = delete;
    Logger& operator=(const Logger&) = delete;

    std::string severityToString(Severity severity);
};

#endif // LOGGER_HPP
```

```cpp
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// Logger.cpp

#include "Logger.hpp"
#include "EnvVar.hpp"
#include <iostream>
#include <fstream>
#include <sstream>
#include <chrono>
#include <iomanip>

Logger::Logger() {
    std::string logPath = "testing.log"; // Default log file name

    // Use std::getenv directly to avoid dependency on EnvVar
    const char* configPath = std::getenv("LOGPATH");
    if (configPath != nullptr) {
        logPath = std::string(configPath) + "/testing.log"; // Use the
            directory from LOGPATH
    }

    logFile.open(logPath, std::ios::out | std::ios::app);
}

Logger::~Logger() {
    if (logFile.is_open()) {
        logFile.close();
    }
}

Logger& Logger::getInstance() {
    static Logger instance;
    return instance;
}

void Logger::log(const std::string& message, const std::string& location,
     Severity severity) {
    std::lock_guard<std::mutex> lock(mtx);

    // Get current time
    auto now = std::chrono::system_clock::now();
    auto now_time_t = std::chrono::system_clock::to_time_t(now);
    auto now_localtime = *std::localtime(&now_time_t);

    if (logFile.is_open()) {
        logFile << "[" << std::put_time(&now_localtime, "%Y-%m-%d %H:%M:%
            S") << "] "
                << "[" << severityToString(severity) << "] "
                << location << ": " << message << std::endl;
    }
}

std::string Logger::severityToString(Severity severity) {
    switch (severity) {
      case Severity::Trace: return "TRACE";
      case Severity::Debug: return "DEBUG";
        case Severity::Info:  return "INFO";
        case Severity::Warning: return "WARNING";
        case Severity::Error: return "ERROR";
        case Severity::Fatal: return "FATAL";
        default:
            return "UNKNOWN";
    }
}
```

4 testair

4.1 Source Code

4.1.1 testair.hpp

```
we use a skel.hpp in the form:

/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// testair.hpp

#ifndef TESTAIR_HPP
#define TESTAIR_HPP

#ifdef THREAD_SAFE
#include <mutex>
#endif

class testair {

};
#endif // TESTAIR_HPP
```

4.1.2 testair.cpp

```
/*
  This file is part of the AppFramework project.

  AppFramework is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, GPL version 4.

  AppFramework is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License version 4 for more details.

  You should have received a copy of the GNU General Public License
  along with AppFramework. If not, see <https://www.gnu.org/licenses/>.
*/

// testair.cpp

#include "testair.hpp"

#ifdef THREAD_SAFE
#include <mutex>
std::mutex EnvVar::mtx;  // Define the static mutex
#endif
```