Program Listing for File app_worker.hpp

Return to documentation for file (include/holoscan/core/app_worker.hpp)

/* * SPDX-FileCopyrightText: Copyright (c) 2023 NVIDIA CORPORATION & AFFILIATES. All rights reserved. * SPDX-License-Identifier: Apache-2.0 * * Licensed under the Apache License, Version 2.0 (the "License"); * you may not use this file except in compliance with the License. * You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License. */ #ifndef HOLOSCAN_CORE_APP_WORKER_HPP #define HOLOSCAN_CORE_APP_WORKER_HPP #include <any> #include <memory> #include <mutex> #include <queue> #include <string> #include <unordered_map> #include <vector> #include "holoscan/core/common.hpp" #include "holoscan/core/graph.hpp" namespace holoscan { *// Forward declarations* struct ConnectionItem; enum class AppWorkerTerminationCode { kSuccess, kCancelled, kFailure, }; struct AppWorkerTerminationStatus { std::string worker_id; AppWorkerTerminationCode error_code; }; class AppWorker { public: explicit AppWorker(Application* app); virtual ~AppWorker(); enum class WorkerMessageCode { kExecuteFragments, kNotifyWorkerExecutionFinished, kTerminateWorker, }; struct WorkerMessage { WorkerMessageCode code; std::any data; }; CLIOptions* options(); std::vector<FragmentNodeType>& target_fragments(); FragmentGraph& fragment_graph(); service::AppWorkerServer* server(std::unique_ptr<service::AppWorkerServer>&& server); service::AppWorkerServer* server(); bool execute_fragments(std::unordered_map<std::string, std::vector<std::shared_ptr<holoscan::ConnectionItem>>>& name_connection_list_map); bool terminate_scheduled_fragments(); void submit_message(WorkerMessage&& message); void process_message_queue(); private: friend class service::AppWorkerServer; std::vector<FragmentNodeType> get_target_fragments(FragmentGraph& fragment_graph); Application* app_ = nullptr; CLIOptions* options_ = nullptr; std::unique_ptr<service::AppWorkerServer> worker_server_; FragmentGraph* fragment_graph_ = nullptr; std::vector<FragmentNodeType> target_fragments_;

```
std::vector<FragmentNodeType> scheduled_fragments_; bool
need_notify_execution_finished_ = false; AppWorkerTerminationCode
termination_code_ = AppWorkerTerminationCode::kSuccess; std::mutex
message_mutex_; std::queue<WorkerMessage> message_queue_; }; } // namespace
holoscan #endif/* HOLOSCAN_CORE_APP_WORKER_HPP */
```