Program Listing for File inference_processor.hpp

Return to documentation for file (`include/holoscan/operators/inference_processor/inference_processor.hpp`)

```
/* * SPDX-FileCopyrightText: Copyright (c) 2022-2024 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_OPERATORS_INFERENCE_PROCESSOR_INFERENCE_PROCESSOR_HPP
#define
HOLOSCAN_OPERATORS_INFERENCE_PROCESSOR_INFERENCE_PROCESSOR_HPP
#include <map> #include <memory> #include <string> #include <vector> #include
"holoscan/core/io_context.hpp" #include "holoscan/core/io_spec.hpp" #include
"holoscan/core/operator.hpp" #include "holoscan/core/operator_spec.hpp"
#include "holoscan/utils/cuda_stream_handler.hpp" #include "holoinfer.hpp"
#include "holoinfer_buffer.hpp" #include "holoinfer_utils.hpp" namespace HoloInfer
= holoscan::inference; namespace holoscan::ops { class InferenceProcessorOp :
public holoscan::Operator { public:
HOLOSCAN_OPERATOR_FORWARD_ARGS(InferenceProcessorOp)
InferenceProcessorOp() = default; void setup(OperatorSpec& spec) override; void
initialize() override; void start() override; void compute(InputContext& op_input,
OutputContext& op_output, ExecutionContext& context) override; struct DataMap {
DataMap() = default; explicit operator bool() const noexcept { return
!mappings_.empty(); } void insert(const std::string& key, const std::string& value) {
mappings_[key] = value; } std::map<std::string, std::string> get_map() const { return
mappings_; } std::map<std::string, std::string> mappings_; }; struct DataVecMap {
DataVecMap() = default; explicit operator bool() const noexcept { return
!mappings_.empty(); } void insert(const std::string& key, const
std::vector<std::string>& value) { for (const auto& val : value)
mappings_[key].push_back(val); } std::map<std::string, std::vector<std::string>>
get_map() const { return mappings_; } std::map<std::string, std::vector<std::string>>
mappings_; }; private: Parameter<DataVecMap> process_operations_;
Parameter<DataVecMap> processed_map_; Parameter<std::string> config_path_;
Parameter<std::vector<std::string>> in_tensor_names_;
Parameter<std::vector<std::string>> out_tensor_names_;
Parameter<std::shared_ptr<Allocator>> allocator_; Parameter<bool>
input_on_cuda_; Parameter<bool> output_on_cuda_; Parameter<bool>
transmit_on_cuda_; Parameter<std::vector<IOSpec*>> receivers_;
Parameter<std::vector<IOSpec*>> transmitter_; void
conditional_disable_output_port(const std::string& name); // Internal state
std::unique_ptr<HoloInfer::ProcessorContext> holoscan_postprocess_context_;
HoloInfer::DataMap data_per_tensor_; std::map<std::string, std::vector<int>>
dims_per_tensor_; const std::string module_{"Inference Processor Operator"};
CudaStreamHandler cuda_stream_handler_; }; } // namespace holoscan::ops #endif/*
HOLOSCAN_OPERATORS_INFERENCE_PROCESSOR_INFERENCE_PROCESSOR_HPP */
```