Class OutputContext

Table of contents

- Defined in File io_context.hpp

Inheritance Relationships

Derived Type

- `public holoscan::gxf::GXFOutputContext` (Class GXFOutputContext)

Class Documentation

class OutputContext

Class to hold the output context.

This class provides the interface to send data to the output ports of the operator.

Subclassed by holoscan::gxf::GXFOutputContext

Public Types

enum class OutputType

The output data type.

Values:

enumerator kSharedPointer

The message data to send is a shared pointer.

enumerator kGXFEntity

The message data to send is a GXF entity.

enumerator kAny

The message data to send is a std::any.

Public Functions

inline OutputContext(ExecutionContext *execution_context, Operator *op)

Construct a new OutputContext object.

outputs for the OutputContext will be set to op->spec()->outputs()

Parameters

- **execution_context** – The pointer to the execution context.
- **op** – The pointer to the operator that this context is associated with.

inline OutputContext(ExecutionContext *execution_context, Operator *op, std::unordered_map<std::string, std::shared_ptr<IOSpec>> &outputs)

Construct a new OutputContext object.

Parameters

- **execution_context** – The pointer to the execution context.
- **op** – The pointer to the operator that this context is associated with.
- **outputs** – The references to the map of the output specs.

inline ExecutionContext *execution_context() const

Get pointer to the execution context.

Returns

The pointer to the execution context.

inline Operator *op() const

Return the operator that this context is associated with.

Returns

The pointer to the operator.

inline std::unordered_map<std::string, std::shared_ptr<IOSpec>> &outputs() const

Return the reference to the map of the output specs.

Returns

The reference to the map of the output specs.

template<typename DataT, typename =
std::enable_if_t<!holoscan::is_one_of_derived_v<DataT, nvidia::gxf::Entity,
std::any>>>
inline void emit(std::shared_ptr<DataT> &data, const char *name = nullptr)

Send a shared pointer of the message data to the output port with the given name.

The object to be sent must be a shared pointer of the message data and the output port with the given name must exist.

If the operator has a single output port, the output port name can be omitted.

Example:

```
class PingTxOp : public holoscan::ops::GXFOperator { public:
HOLOSCAN_OPERATOR_FORWARD_ARGS_SUPER(PingTxOp,
holoscan::ops::GXFOperator) PingTxOp() = default; void
setup(OperatorSpec& spec) override { spec.output<ValueData>("out"); }
void compute(InputContext&, OutputContext& op_output,
ExecutionContext&) override { auto value =
std::make_shared<ValueData>(7); op_output.emit(value, "out"); } };
```

Template Parameters

DataT – The type of the data to send.

Parameters

- **data** – The shared pointer to the data.
- **name** – The name of the output port.

template<typename DataT, typename = std::enable_if_t<holoscan::is_one_of_derived_v<DataT, nvidia::gxf::Entity>>>
inline void emit(DataT &data, const char *name = nullptr)

Send message data (GXF Entity) to the output port with the given name.

This method is for interoperability with the GXF Codelet.

The object to be sent must be an object with `holoscan::gxf::Entity` type and the output port with the given name must exist.

If the operator has a single output port, the output port name can be omitted.

Example:

```
class PingTxOp : public holoscan::ops::GXFOperator { public: HOLOSCAN_OPERATOR_FORWARD_ARGS_SUPER(PingTxOp, holoscan::ops::GXFOperator) PingTxOp() = default; void setup(OperatorSpec& spec) override { spec.input<holoscan::gxf::Entity>("in"); spec.output<holoscan::gxf::Entity>("out"); } void compute(InputContext& op_input, OutputContext& op_output, ExecutionContext&) override { // The type of `in_message` is 'holoscan::gxf::Entity'. auto in_message = op_input.receive<holoscan::gxf::Entity>("in"); // The type of `tensor` is 'std::shared_ptr<holoscan::Tensor>'. auto tensor = in_message.get<Tensor>(); // Process with 'tensor' here. // ... // Create a new message (Entity) auto out_message = holoscan::gxf::Entity::New(&context); out_message.add(tensor, "tensor"); // Send the processed message. op_output.emit(out_message, "out"); } };
```

Template Parameters

DataT – The type of the data to send. It should be `holoscan::gxf::Entity`.

Parameters

- **data** – The entity object to send (`holoscan::gxf::Entity`).

- **name** – The name of the output port.

template<typename DataT, typename = std::enable_if_t<!holoscan::is_one_of_derived_v<DataT, nvidia::gxf::Entity>>>
inline void emit(DataT data, const char *name = nullptr)

Send the message data (std::any) to the output port with the given name.

This method is for interoperability with arbitrary data types.

The object to be sent can be any type except the shared pointer (std::shared_ptr<T>) or the GXF Entity (holoscan::gxf::Entity) type, and the output port with the given name must exist.

If the operator has a single output port, the output port name can be omitted.

Example:

```
class PingTxOp : public holoscan::ops::GXFOperator { public:
HOLOSCAN_OPERATOR_FORWARD_ARGS_SUPER(PingTxOp,
holoscan::ops::GXFOperator) PingTxOp() = default; void
setup(OperatorSpec& spec) override { spec.input<holoscan::gxf::Entity>
("in"); spec.output<holoscan::gxf::Entity>("out"); } void
compute(InputContext& op_input, OutputContext& op_output,
ExecutionContext&) override { // The type of `in_message` is
'holoscan::gxf::Entity'. auto in_message =
op_input.receive<holoscan::gxf::Entity>("in"); // The type of `tensor` is
'std::shared_ptr<holoscan::Tensor>'. auto tensor = in_message.get<Tensor>
(); // Process with 'tensor' here. // ... // Create a new message (Entity) auto
out_message = holoscan::gxf::Entity::New(&context);
out_message.add(tensor, "tensor"); // Send the processed message.
op_output.emit(out_message, "out"); } };
```

Template Parameters

DataT – The type of the data to send. It can be any type except the shared pointer (std::shared_ptr<T>) or the GXF Entity (holoscan::gxf::Entity) type.

Parameters

- **data** – The entity object to send (as `std::any`).
- **name** – The name of the output port.

inline void emit(holoscan::TensorMap &data, const char *name = nullptr)

Protected Functions

inline virtual void emit_impl(std::any data, const char *name = nullptr, OutputType out_type = OutputType::kSharedPointer)

The implementation of the `emit` method.

Depending on the type of the data, this method wraps the data with a message and sends it to the output port with the given name.

Parameters

- **data** – The data to send.
- **name** – The name of the output port.
- **out_type** – The type of the message data.

Protected Attributes

ExecutionContext *execution_context_ = nullptr

The execution context that is associated with.

Operator *op_ = nullptr

The operator that this context is associated with.

std::unordered_map<std::string, std::shared_ptr<IOSpec>> &outputs_

The outputs.