Class Executor

Table of contents

- Defined in File executor.hpp

Inheritance Relationships

Derived Type

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

Class Documentation

class Executor

Base class for all executors.

An Executor that manages the execution of a Fragment on a physical node. The framework provides a default Executor that uses a GXF Scheduler to execute an Application.

Subclassed by holoscan::gxf::GXFExecutor

Public Functions

Executor() = delete

inline explicit Executor(Fragment *fragment)

> Construct a new Executor object.
>
> Parameters
>
> **fragment** – The pointer to the fragment of the executor.

virtual ~Executor() = default

inline virtual void run(OperatorGraph &graph)

> Run the graph.
>
> Parameters
>
> **graph** – The reference to the graph.

inline virtual std::future<void> run_async(OperatorGraph &graph)

Run the graph asynchronously.

Parameters

graph – The reference to the graph.

Returns

The future object.

inline virtual void interrupt()

Interrupt the execution.

inline void fragment(Fragment *fragment)

Set the pointer to the fragment of the executor.

Parameters

fragment – The pointer to the fragment of the executor.

inline Fragment *fragment()

Get a pointer to Fragment object.

Returns

The Pointer to Fragment object.

inline virtual void context(void *context)

Set the context.

Parameters

context – The context.

inline void *context()

Get the context.

Returns

The context.

inline void context_uint64(uint64_t context)

inline uint64_t context_uint64()

inline virtual std::shared_ptr<ExtensionManager> extension_manager()

Get the extension manager.

Returns

The shared pointer of the extension manager.

inline void exception(const std::exception_ptr &e)

Set the exception.

This method is called by the framework to store the exception that occurred during the execution of the fragment. If the exception is set, this exception is rethrown by the framework after the execution of the fragment.

Parameters

e – The exception to store.

inline const std::exception_ptr &exception()

Get the stored exception.

This method is called by the framework to get the stored exception that occurred during the execution of the fragment. If the exception is set, this exception is rethrown by the framework after the execution of the fragment.

Returns

The reference to the stored exception.

Protected Functions

inline virtual bool initialize_fragment()

Initialize the fragment_ in this Executor.

This method is called by run() to initialize the fragment and the graph of operators in the fragment before execution.

Returns

true if fragment initialization is successful. Otherwise, false.

inline virtual bool initialize_operator(Operator *op)

Initialize the given operator.

This method is called by Operator::initialize() to initialize the operator.

Depending on the type of the operator, this method may be overridden to initialize the operator. For example, the default executor (GXFExecutor) initializes the operator using the GXF API and sets the operator's ID to the ID of the GXF codelet.

Parameters

op – The pointer to the operator.

Returns

true if the operator is initialized successfully. Otherwise, false.

inline virtual bool initialize_scheduler(Scheduler *sch)

Initialize the given scheduler.

This method is called by Scheduler::initialize() to initialize the operator.

Depending on the type of the scheduler, this method may be overridden to initialize the scheduler. For example, the default executor (GXFExecutor) initializes the scheduler using the GXF API and sets the operator's ID to the ID of the GXF scheduler.

Parameters

sch – The pointer to the scheduler.

Returns

true if the scheduler is initialized successfully. Otherwise, false.

inline virtual bool initialize_network_context(NetworkContext *network_context)

Initialize the given network context.

This method is called by NetworkContext::initialize() to initialize the operator.

Depending on the type of the network context, this method may be overridden to initialize the network context. For example, the default executor (GXFExecutor) initializes the network context using the GXF API and sets the operator's ID to the ID of the GXF network context.

Parameters

network_context – The pointer to the network context.

Returns

true if the network context is initialized successfully. Otherwise, false.

inline virtual bool add_receivers(const std::shared_ptr<Operator> &op, const std::string &receivers_name, std::vector<std::string> &new_input_labels, std::vector<holoscan::IOSpec*> &iospec_vector)

Add the receivers as input ports of the given operator.

This method is to be called by the Fragment::add_flow() method to support for the case where the destination input port label points to the parameter name of the downstream operator, and the parameter type is 'std::vector<holoscan::IOSpec*>'. This finds a parameter with with 'std::vector<holoscan::IOSpec*>' type and create a new input port with a specific label ('<parameter name>:<index>'. e.g, 'receivers:0').

Parameters

- **op** – The reference to the shared pointer of the operator.

- **receivers_name** – The name of the receivers whose parameter type is ‘std::vector<holoscan::IOSpec*>’.

- **new_input_labels** – The reference to the vector of input port labels to which the input port labels are added. In the case of multiple receivers, the input port label is updated to ‘<parameter name>:<index>’ (e.g. ‘receivers’ => ‘receivers:<index>’).

- **iospec_vector** – The reference to the vector of IOSpec pointers.

Returns

true if the receivers are added successfully. Otherwise, false.

Protected Attributes

Fragment *fragment_ = nullptr

The fragment of the executor.

void *context_ = nullptr

The context.

std::shared_ptr<ExtensionManager> extension_manager_

The extension manager.

std::exception_ptr exception_

The stored exception.

Friends

friend class Fragment

friend class Operator

friend class Scheduler

friend class NetworkContext