Class GXFOperator

Table of contents

- Defined in File gxf_operator.hpp

Inheritance Relationships

Base Type

- `public holoscan::Operator` (Class Operator)

Derived Type

- `public holoscan::ops::GXFCodeletOp` (Class GXFCodeletOp)

Class Documentation

class GXFOperator : public holoscan::Operator

Subclassed by holoscan::ops::GXFCodeletOp

Public Functions

template<typename ArgT, typename ...ArgsT, typename = std::enable_if_t<!std::is_base_of_v<holoscan::Operator, std::decay_t<ArgT>> && (std::is_same_v<holoscan::Arg, std::decay_t<ArgT>> || std::is_same_v<holoscan::ArgList, std::decay_t<ArgT>> || std::is_base_of_v<holoscan::Condition, typename holoscan::type_info<ArgT>::derived_type> || std::is_base_of_v<holoscan::Resource, typename holoscan::type_info<ArgT>::derived_type>)>>
inline explicit GXFOperator(ArgT &&arg, ArgsT&&... args)

Construct a new GXFOperator object.

Parameters

args – The arguments to be passed to the operator.

inline GXFOperator()

virtual void initialize() override

Initialize the GXF operator.

This function is called when the fragment is initialized by Executor::initialize_fragment().

This sets the operator type to `holoscan::Operator::OperatorType::kGXF`.

virtual const char *gxf_typename() const = 0

Get the type name of the GXF component.

The returned string is the type name of the GXF component and is used to create the GXF component.

Example: “nvidia::holoscan::AJASource”

Returns

The type name of the GXF component.

inline gxf_context_t gxf_context() const

Get the GXF context object.

Returns

The GXF context object.

inline void gxf_eid(gxf_uid_t gxf_eid)

Set GXF entity ID.

Parameters

gxf_eid – The GXF entity ID.

inline gxf_uid_t gxf_eid() const

Get the GXF entity ID.

Returns

The GXF entity ID.

inline void gxf_cid(gxf_uid_t gxf_cid)

Set the GXF component ID.

Parameters

gxf_cid – The GXF component ID.

inline gxf_uid_t gxf_cid() const

Get the GXF component ID.

Returns

The GXF component ID.

Public Static Functions

template<typename typeT>
static inline void register_converter()

Register the argument setter and the GXF parameter adaptor for the given type.

If the GXF operator has an argument with a custom type, both the argument setter and GXF parameter adaptor must be registered using this method.

The argument setter is used to set the value of the argument from the YAML configuration, and the GXF parameter adaptor is used to set the value of the GXF parameter from the argument value in `YAML::Node` object.

This method can be called in the initialization phase of the operator (e.g., `initialize()`). The example below shows how to register the argument setter for the custom type (`Vec3`):

```
void MyGXFOp::initialize() { register_converter<Vec3>();
holoscan::ops::GXFOperator::initialize(); }
```

It is assumed that `YAML::convert<T>::encode` and `YAML::convert<T>::decode` are implemented for the given type. You

need to specialize the `YAML::convert<>` template class.

For example, suppose that you had a `Vec3` class with the following members:

```
struct Vec3 { // make sure you have overloaded operator==() for the comparison double x, y, z; };
```

You can define the `YAML::convert<Vec3>` as follows in a '.cpp' file:

```
namespace YAML { template<> struct convert<Vec3> { static Node encode(const Vec3& rhs) { Node node; node.push_back(rhs.x); node.push_back(rhs.y); node.push_back(rhs.z); return node; } static bool decode(const Node& node, Vec3& rhs) { if(!node.IsSequence() || node.size() != 3) { return false; } rhs.x = node[0].as<double>(); rhs.y = node[1].as<double>(); rhs.z = node[2].as<double>(); return true; } }; }
```

Please refer to the yaml-cpp documentation for more details.

Template Parameters

typeT – The type of the argument to register.

Protected Functions

virtual gxf_uid_t add_codelet_to_graph_entity() override

This method is invoked by 'GXFExecutor::initialize_operator(Operator* op)' during the initialization of the operator. By overriding this method, additional setup tasks are performed for the operator, including:

- Initializing the `spec_` object with the codelet's parameters.

Returns

The codelet component id corresponding to GXF codelet.

virtual void set_parameters() override

This method is invoked at the end of 'GXFExecutor::initialize_operator(Operator* op)' during the initialization of the operator. By overriding this method, we can modify how GXF Codelet's parameters are set from the arguments.

Protected Attributes

gxf_context_t gxf_context_ = nullptr

The GXF context.

gxf_uid_t gxf_eid_ = 0

GXF entity ID.

gxf_uid_t gxf_cid_ = 0

The GXF component ID.

nvidia::gxf::Handle<nvidia::gxf::Codelet> codelet_handle_

The codelet handle.

std::string gxf_typename_ = "unknown_gxf_typename"

The GXF type name (used for GXFCodeletOp)

Protected Static Functions

template<typename typeT>
static inline void register_parameter_adaptor()

Register the GXF parameter adaptor for the given type.

Please refer to the documentation of `register_converter()` for more details.

Template Parameters

typeT – The type of the argument to register.