Version: 4.0.1

Pass additional arguments to your Nodes

In every single example we explored so far, we were "forced" to provide a constructor with the following signature

```
MyCustomNode(const std::string& name, const NodeConfig& config);
```

In some cases, it is desirable to pass additional arguments, parameters, pointers, references, etc, to the constructor of our class.

> **CAUTION**
>
> Some people use blackboards to do that. **Don't**.

We will just use the word *"arguments"* for the rest of the tutorial.

Even if, theoretically, these arguments **could** be passed using Input Ports, that would be the wrong way to do it if:

- The arguments are known at *deployment-time* (when building the tree).
- The arguments don't change at *run-time*.
- The arguments don't need to be set from the XML.

If all these conditions are met, using ports or the blackboard is highly discouraged.

Add arguments to your constructor (recommended)

Consider the following custom node called Action_A.

We want to pass three additional arguments; they can be arbitrarily complex objects, you are not limited to built-in types.

```
// Action_A has a different constructor than the default one.
class Action_A: public SyncActionNode
{

public:
    // additional arguments passed to the constructor
    Action_A(const std::string& name, const NodeConfig& config,
             int arg_int, std::string arg_str ):
        SyncActionNode(name, config),
        _arg1(arg_int),
        _arg2(arg_str) {}

    // this example doesn't require any port
    static PortsList providedPorts() { return {}; }

    // tick() can access the private members
    NodeStatus tick() override;

private:
    int _arg1;
    std::string _arg2;
};
```

Registering this node and passing the known arguments is as easy as:

```
BT::BehaviorTreeFactory factory;
factory.registerNodeType<Action_A>("Action_A", 42, "hello world");

// If you prefer to specify the template parameters
// factory.registerNodeType<Action_A, int , std::string>("Action_A", 42,
"hello world");
```

Use an "initialize" method

If, for any reason, you need to pass different values to the individual instances of a Node type, you may want to consider this other pattern:

```
class Action_B: public SyncActionNode
{

public:
    // The constructor looks as usual.
    Action_B(const std::string& name, const NodeConfig& config):
        SyncActionNode(name, config) {}

    // We want this method to be called ONCE and BEFORE the first tick()
    void initialize( int arg_int, const std::string& arg_str_ )
    {
        _arg1 = arg_int;
        _arg2 = arg_str_;
    }

    // this example doesn't require any port
    static PortsList providedPorts() { return {}; }

    // tick() can access the private members
    NodeStatus tick() override;

private:
    int _arg1;
    std::string _arg2;
};
```

The way we register and initialize Action_B is different:

```
BT::BehaviorTreeFactory factory;

// Register as usual, but we still need to initialize
factory.registerNodeType<Action_B>( "Action_B" );

// Create the whole tree. Instances of Action_B are not initialized yet
auto tree = factory.createTreeFromText(xml_text);

// visitor will initialize the instances of
auto visitor = [](TreeNode* node)
{
  if (auto action_B_node = dynamic_cast<Action_B*>(node))
  {
```

```
    action_B_node->initialize(69, "interesting_value");
  }
};
// Apply the visitor to ALL the nodes of the tree
tree.applyVisitor(visitor);
```

Edit this page