

Writing a Simple Procedure in Karabo (How to Macro)

Anna Klimovskaia / Gabriele Giovanetti
European XFEL - Controls Group

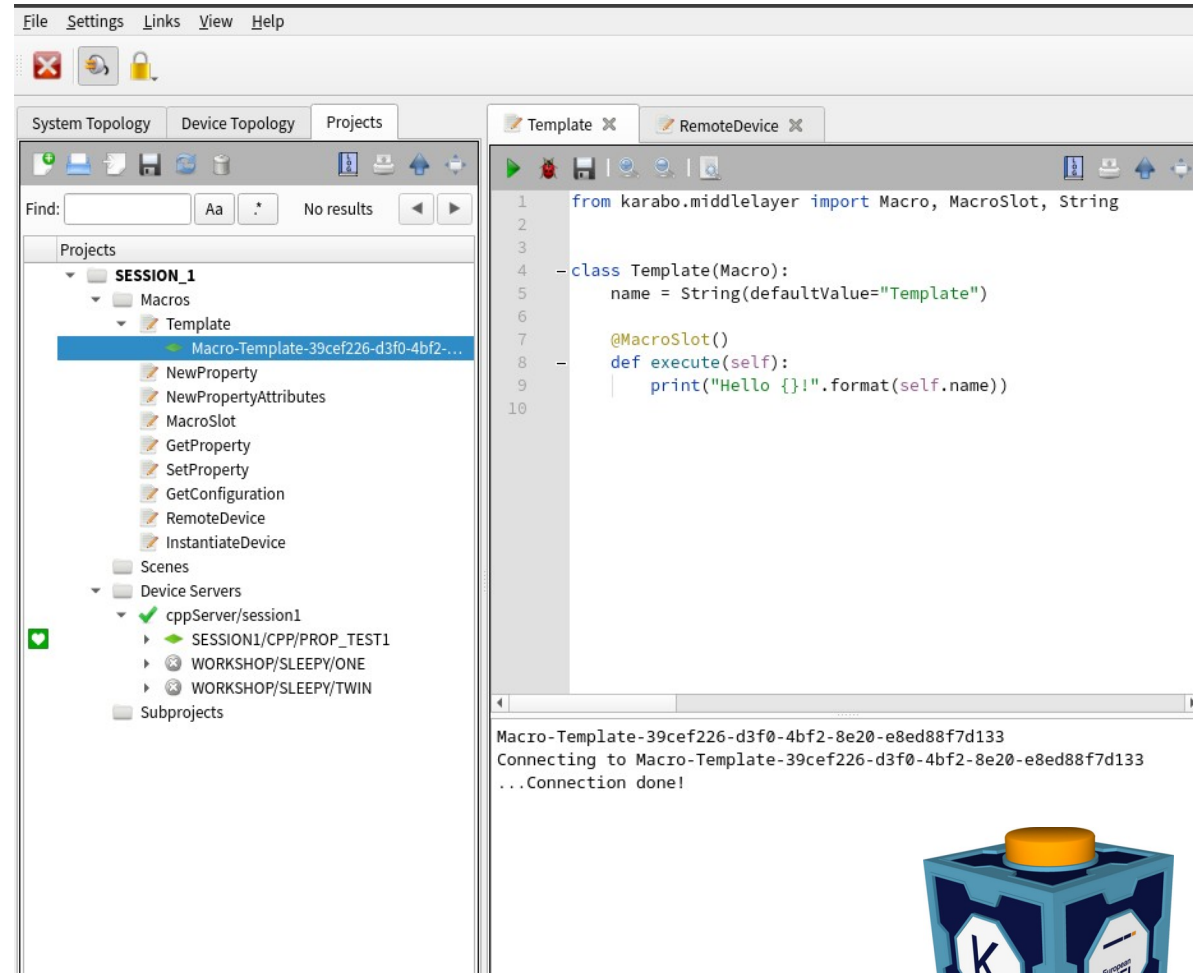
NOBUGS2024 - Grenoble, 26.9.2024



Macro Fundamentals

The Karabo framework provides scripting facilities which enable Instrument scientists to develop so-called "macros".

Macros are short python scripts that give the flexibility to write ad-hoc code by Karabo users directly in a karabo GUI.



The screenshot displays the Karabo GUI interface. On the left, a project tree shows a session named 'SESSION_1' with a sub-folder 'Macros' containing a 'Template' macro. The 'Template' macro is selected, and its contents are shown in the main editor window. The macro code is as follows:

```

1 from karabo.middlayer import Macro, MacroSlot, String
2
3
4 -class Template(Macro):
5     name = String(defaultValue="Template")
6
7     @MacroSlot()
8     def execute(self):
9         print("Hello {}".format(self.name))
10

```

Below the editor, a console window shows the output of the macro execution:

```

Macro-Template-39cef226-d3f0-4bf2-8e20-e8ed88f7d133
Connecting to Macro-Template-39cef226-d3f0-4bf2-8e20-e8ed88f7d133
...Connection done!

```



Macro Benefits

- Macros can be written quickly by Karabo users
- Macros can be saved as part of project configurations – covers recurring task and calculations
- Macros may be executed from the GUI
- Simplified syntax, wrt. To KARabo middle-layer devices.
- Interaction with karabo devices

Macro Limitation

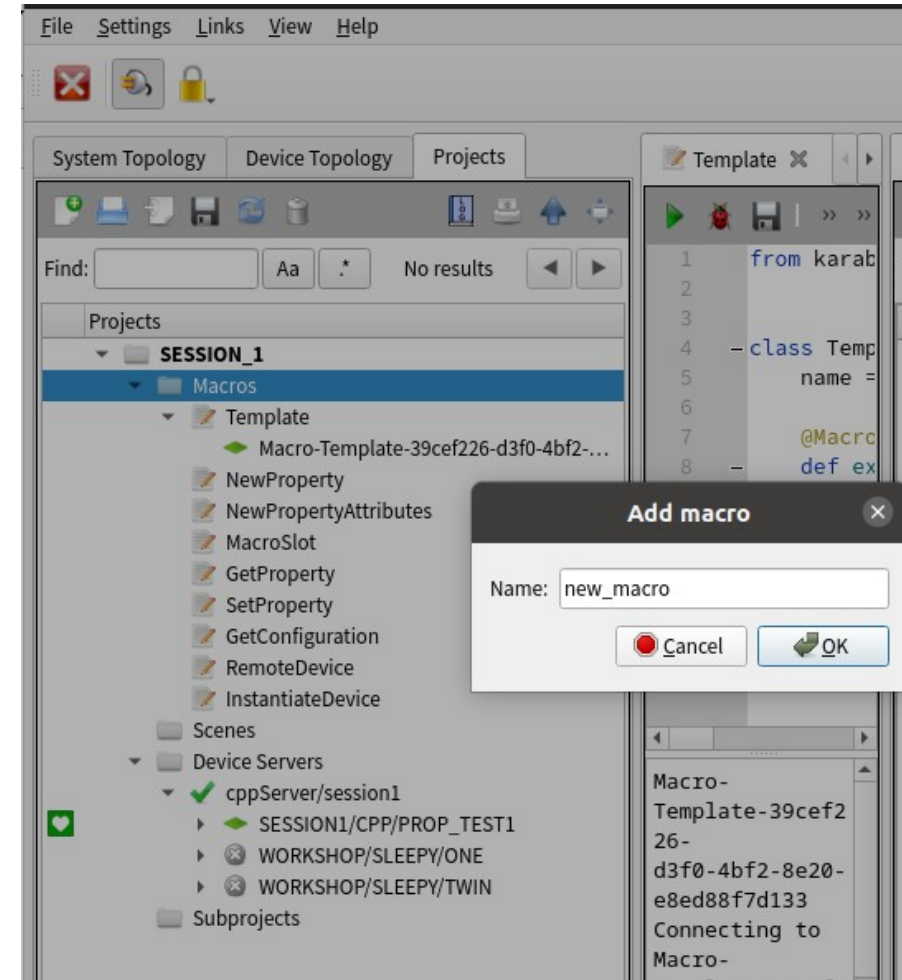
- Macros by concept should be kept simple
- Should not have state machine (Only ACTIVE and PASSIVE automatically set)
- Can be canceled
- Macros utilize the Karabo Middlelayer API interface. However, macros are only suitable for simple use cases
- There is no version control and limited support
- No configuration logging

Macro Caveats

- A misbehaving macro could screw up all other macros running on the same macro server.
 - Test your macros in the development server when it is available ('debug' button).
 - Please follow good practices and use advised programming patterns.

How To Create Macro

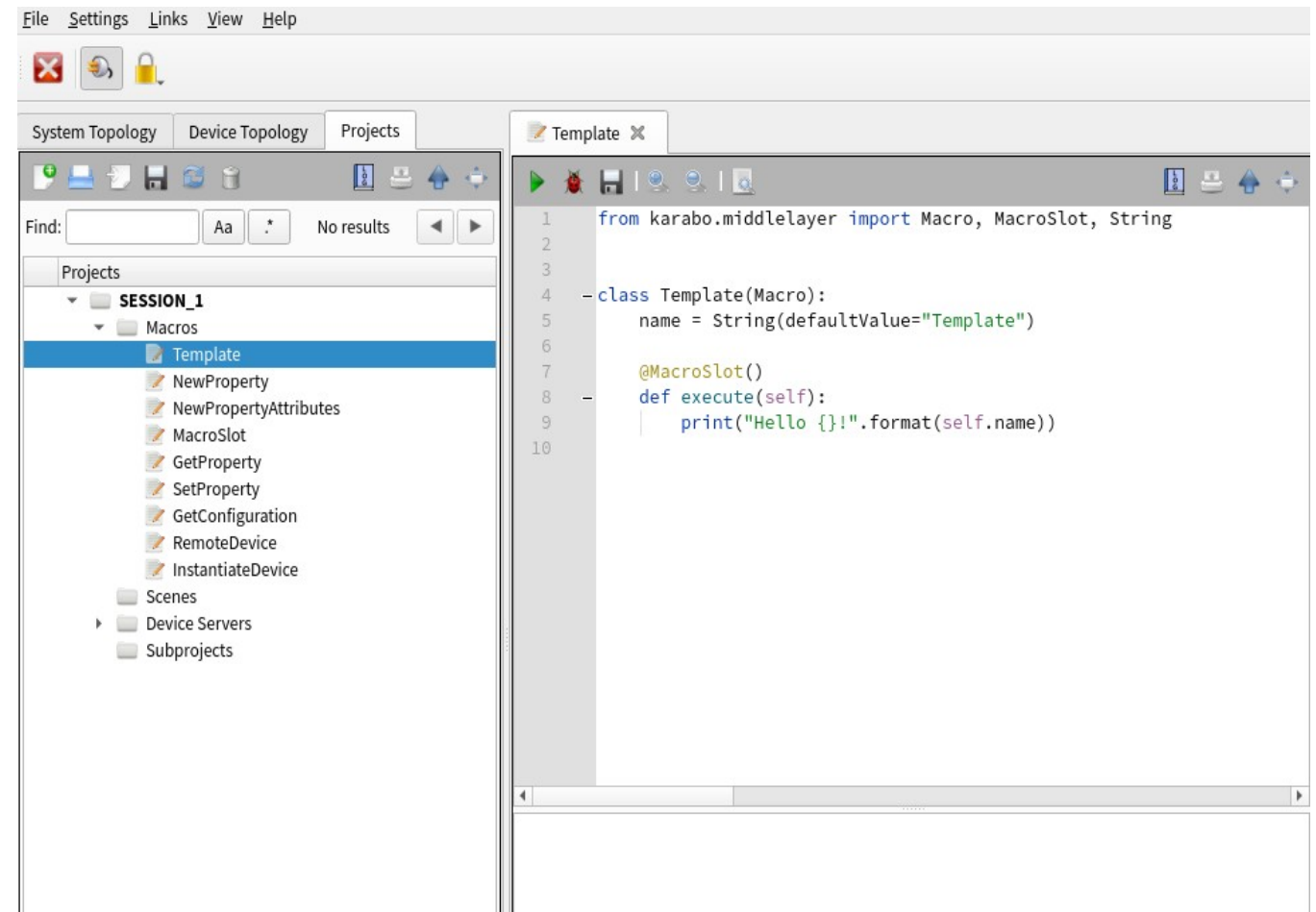
- Connect to karabo (admin, localhost:44444)
- Open project
- Select the macro folder
- Right-click and then click Add macro
- Fill the dialog with a macro-name
- As a result - new macro is added to the project with default content



Macro Content

- Import from middlelayer api
- A Macro class
- Properties (Karabo descriptors), e.g. String
- MacroSlots to execute assigned functions

Note: A Macro always comes with a Cancel Slot - no need to define it



```
File Settings Links View Help
System Topology Device Topology Projects
Find: No results
Projects
  SESSION_1
    Macros
      Template
      NewProperty
      NewPropertyAttributes
      MacroSlot
      GetProperty
      SetProperty
      GetConfiguration
      RemoteDevice
      InstantiateDevice
    Scenes
    Device Servers
    Subprojects

1 from karabo.middlelayer import Macro, MacroSlot, String
2
3
4 -class Template(Macro):
5     name = String(defaultValue="Template")
6
7     @MacroSlot()
8     def execute(self):
9         print("Hello {}".format(self.name))
10
```

Macro Content

- Import from middlelayer api
- A Macro class
- Properties (Karabo descriptors), e.g. String
- MacroSlots to execute assigned functions

Note: A Macro always comes with a Cancel Slot - no need to define it

The screenshot displays the Karabo Configuration Editor interface. On the left, a tree view shows the project structure under 'WORKSHOP_24_Session1', with 'Macro-Template-1c75887b-9f2d-4e8e-...' selected. The main editor shows the Python code for the 'Template' macro class:

```

1 from karabo.middlelayer import Macro, MacroSlot, S
2
3
4 -class Template(Macro):
5     name = String(defaultValue="Template")
6
7     @MacroSlot()
8     def execute(self):
9         print("Hello {}".format(self.name))
10

```

Below the code, a log window shows the connection status: 'Macro-Template-1c75887b-9f2d-4e8e-b8d6-520033b7a93c Connecting to Macro-Template-1c75887b-9f2d-4e8e-b8d6-520033b7a93c ... Connection done!'.

On the right, the 'Configuration Editor' shows a table of properties and their current values on the device:

Property	Current value on device
ClassID	Template
Class version	builtins-2.20.0a5
Karabo version	2.20.0a5
ServerID	karabo/macroServer
Host	cas-lab-sys-conl-2
Process ID	15194
State	PASSIVE
Status	
Alarm condition	none
Locked By	
Last command	
Logger	
Project	WORKSHOP_24_Session1
Module	Template
Current Slot	
Printed output	
Number of prints	0
Available Macros	['macro']
name	Template

At the bottom of the configuration editor, there are buttons for 'Cancel', 'execute', and 'Shutdown instance', along with an 'Apply all' checkbox.

Code Style

- Class Definitions – CamelCase starting with capital letter
- Class Properties - camelCase, whereas non-exposed variables should have_underscores
- Slots are camelCase, methods have_underscores.
- It is preferred to check for conditions to be correct rather than using exceptions
- Use Double and NOT Float

```
class MyMacro(Macro):  
    pass  
  
class CallDRC(Macro):  
    pass
```

Code Style

- Class Definitions – CamelCase starting with capital letter
- Class Properties - camelCase, whereas non-exposed variables should have `_underscores`
- Slots are camelCase, methods have `_underscores`.
- It is preferred to check for conditions to be correct rather than using exceptions
- Use Double and NOT Float

```
name = String(displayedName="Name")

someOtherString = String(
    displayedName="Other string",
    defaultValue="Hello",
    accessMode=AccessMode.READONLY)

valid_ids = ["44eab", "ff64d"]
```

Code Style

- Class Definitions – CamelCase starting with capital letter
- Class Properties - camelCase, whereas non-exposed variables should have_underscores
- Slots are camelCase, methods have_underscores.
- It is preferred to check for conditions to be correct rather than using exceptions
- Use Double and NOT Float

```
@MacroSlot(displayedName="Execute")
def executeMe(self):
    | await self.some_action()

def some_action(self):
    | pass
```

Code Style

- Class Definitions – CamelCase starting with capital letter
- Class Properties - camelCase, whereas non-exposed variables should have_underscores
- Slots are camelCase, methods have_underscores.
- It is preferred to check for conditions to be correct rather than using exceptions
- Use Double and NOT Float

Therefore, the following `is` discouraged:

```
def execute_action(self):  
    try:  
        await self.px.move()  
    except:  
        pass
```

But rather:

```
def execute_action(self):  
    if self.px.state not in {State.ERROR, State.MOVING}:  
        await self.px.move()  
    else:  
        pass
```

Code Examples

- Template - default macro content
- NewProperty - add new property
- NewPropertyAttributes - property attributes
- MacroSlot - add slot (button)
- GetProperty - get property from another device
- SetProperty - set new value to another device
- GetConfiguration - get full configuration
- RemoteDevice - monitor remote device
- InstantiateDevice - instantiate device with required configuration

Template

- Import
- Instantiate on macro server
- Instantiate on development macro server
- Slot is a button
- Execution output

```
1 from karabo.middlayer import Macro, MacroSlot, String
2
3
4 - class Template(Macro):
5     name = String(defaultValue="Template")
6
7     @MacroSlot()
8     - def execute(self):
9         | print("Hello {}!".format(self.name))
10
```

NewProperty

- Different types
- Properties order

```
1 from karabo.middlelayer import Int32, Double, Macro
2
3
4 -class NewProperty(Macro):
5
6     currentVoltage = Double(defaultValue=0.0)
7     steps = Int32()
```

NewPropertyAttributes

- Import
- List of attributes

```
1 - from karabo.middlelayer import (  
2     AccessMode, Assignment, Double, Macro, MetricPrefix, Unit  
3 )  
4  
5  
6 - class NewPropertyAttributes(Macro):  
7  
8     - currentVoltage = Double(  
9         displayedName="Current Voltage",  
10        description="Current Voltage of pico motor",  
11        defaultValue=0.0,  
12        assignment=Assignment.OPTIONAL,  
13        accessMode=AccessMode.READONLY,  
14        unitSymbol=Unit.VOLT,  
15        metricPrefixSymbol=MetricPrefix.KILO)  
16
```


Task1 - NewPropertyAttributes

- Add new Int32 property
 - Don't forget to import proper type
 - Change property key
 - Set new default value

```
1 - from karabo.middlelayer import (  
2     AccessMode, Assignment, Double, Int32, Macro, MetricPrefix, Unit  
3 )  
4  
5  
6 - class NewPropertyAttributes(Macro):  
7  
8     currentVoltage = Double(  
9         displayedName="Current Voltage",  
10        description="Current Voltage of pico motor",  
11        defaultValue=0.0,  
12        assignment=Assignment.OPTIONAL,  
13        accessMode=AccessMode.READONLY,  
14        unitSymbol=Unit.VOLT,  
15        metricPrefixSymbol=MetricPrefix.KILO)  
16  
17 - newInt = Int32(  
18     displayedName="int32",  
19     defaultValue=1)  
20
```

MacroSlot

- Import
- Execution output

```
1 from karabo.middlelayer import Macro, MacroSlot, String
2
3
4 - class MacroSlot(Macro):
5
6     name = String(defaultValue="Macroslot")
7
8     @MacroSlot()
9     - def printName(self):
10         |     print(f"Hello {self.name}!")
```

Task2 - MacroSlot

- Calculate new value and set to macro property
 - Update import
 - Change displayed name
 - Add new properties
 - Set new value x2

```
1 from karabo.middlelayer import AccessMode, Int8, Macro, MacroSlot, String
2
3
4 -class MacroSlot(Macro):
5
6     name = String(defaultValue="Macroslot")
7
8     @MacroSlot()
9     -def printName(self):
10         | print(f"Hello {self.name}!")
11
12     newInt = Int8()
13
14     @MacroSlot()
15     -def multiplication(self):
16         | self.calculatedInt = self.newInt*2
17
18     calculatedInt = Int8(accessMode=AccessMode.READONLY)
19
```

GetProperty

- Import
- GetProperties input parameters

```
1 from karabo.middlelayer import Macro, MacroSlot, String, getProperties
2
3
4 -class GetProperty(Macro):
5
6     @MacroSlot(displayedName="Read Value")
7     - def readString(self):
8     -     self.name = getProperties("SESSION1/PROP_TEST1",
9     -                             ["stringProperty", "int8Property"])["stringProperty"]
10
11     name = String(displayedName="Value on Device")
12
```

SetProperty

- Import
- Scene Set_Value

```
1 from karabo.middlelayer import Int8, Macro, MacroSlot, setWait
2
3
4 - class SetProperty(Macro):
5     number = Int8(defaultValue=1)
6
7     @MacroSlot()
8     - def setInt(self):
9         new_value = self.number.value * 2
10        setWait("SESSION1/CPP/PROP_TEST1", "int8Property", new_value)
11
```

Task3 - SetProperty

- Add new slot to set any other property on remote device
 - Don't forget to import proper type
 - Change property key
 - Change set property key

```
1 from karabo.middlayer import Int8, Macro, MacroSlot, setWait, String
2
3
4 - class SetProperty(Macro):
5     number = Int8(defaultValue=1)
6
7     @MacroSlot()
8     - def setInt(self):
9         new_value = self.number.value * 2
10        setWait("SESSION1/CPP/PROP_TEST1", "int8Property", new_value)
11
12        newString = String()
13
14        @MacroSlot()
15        - def setStr(self):
16            setWait("SESSION1/CPP/PROP_TEST1", "stringProperty", self.newString)
17
```

GetConfiguration

- Import
- getConfiguration input parameters

```
1 from karabo.middlelayer import Int32, Macro, MacroSlot, getConfiguration
2
3
4 -class GetConfiguration(Macro):
5
6     @MacroSlot()
7     -def execute(self):
8         config = getConfiguration("SESSION1/CPP/PROP_TEST1")
9         print(config.keys())
10        self.testInt = config["int32Property"]
11
12    testInt = Int32()
13
```

RemoteDevice

- Import
- RemoveDevice input parameters
- WaitUntilNew
- Change monitoring property
- Note the optional async/await syntax

```
RemoteDevice
1 - from karabo.middlelayer import (
2     Macro, MacroSlot, String, connectDevice, waitUntilNew)
3
4
5
6 - class RemoteDevice(Macro):
7
8     propertyToMonitor = String(
9         displayedName="Monitoring property",
10        defaultValue="int8Property")
11
12    @MacroSlot(displayedName="Start Monitoring")
13    - async def startMonitor(self):
14        device_proxy = await connectDevice('SESSION1/PPP/PROP_TEST1')
15        while True:
16            current_monitor_prop = getattr(device_proxy,
17                self.propertyToMonitor)
18            print(f"Received {current_monitor_prop.value}")
19            await waitUntilNew(current_monitor_prop)
20
```


InstantiateDevice

- Import
- Sleep
- Set several properties on instantiation

```
1 - from karabo.middlelayer import (  
2     instantiate, Hash, Macro, MacroSlot, sleep, String, shutdown  
3 )  
4 import datetime  
5  
6 - class InstantiateDevice(Macro):  
7     name = String(defaultValue="WORKSHOP/SLEEPY/ONE")  
8  
9     @MacroSlot()  
10 - def execute(self):  
11         h = Hash()  
12         current_time = datetime.datetime.now()  
13         h["stringProperty"] = str(current_time)  
14         instantiate('cppServer/session1', 'PropertyTest', self.name, h)  
15         print(f"device {self.name} is instantiated with sting: {current_time}")  
16         sleep(10)  
17         shutdown(self.name)  
18         print(f"device {self.name} shutdown")  
19         sleep(2)  
20
```

Task4 - InstantiateDevice

- Add several properties to initial device configuration
 - Import types
 - Add new property to hash

```
2     instantiate, Hash, Macro, MacroSlot, sleep, String, shutdown
3 )
4 import datetime
5
6 - class InstantiateDevice(Macro):
7     name = String(defaultValue="WORKSHOP/SLEEPY/ONE")
8     newInt = Int8(defaultValue=1)
9
10    @MacroSlot()
11    - def execute(self):
12        h = Hash()
13        current_time = datetime.datetime.now()
14        h["stringProperty"] = str(current_time)
15        h["int8Prtoperty"] = self.newInt
16        instantiate('cppServer/session1', 'PropertyTest', self.name, h)
17        print(f"device {self.name} is instantiated with sting: {current_time}")
18        sleep(10)
19        shutdown(self.name)
20        print(f"device {self.name} shutdown")
21        sleep(2)
22
```