qth_yarp Documentation

Release 0.3.0

Jonathan Heathcote

Jan 15, 2022

Contents

This library implements a simplified API for writing certain types of Qth clients. qth_yarp uses the yarp library to expose a reactive programming style interface to Qth. Clients written using qth_yarp can be just a handful of lines and be entirely free from callbacks. For example, the following converts a Qth temperature property from degrees Celsius to Kelvin, storing the result in a new property:

```
from qth_yarp import get_property, set_property, run_forever

temperature_celsius = get_property("house/temperature", 19.0)
temperature_kelvin = temperature_celsius + 273.15

set_property("house/temperature-in-kelvin",
             temperature_kelvin,
             register=True,
             description="Current temperature in kelvin.")

run_forever()
```

qth_yarp is not intended as a replacement for the lower-level Qth API but rather to complement it. While it makes certain types of clients easier to write, particularly 'if-this-then-that'-style or value-transformation focused clients. It is not suitable for writing clients which maintain complex program state or which need to dynamically watch, subscribe or create properties and events. For these more complex applications, the standard Qth API should be used.

qth_yarp and the low-level Qth API can be used simultaneously within the same client in cases where only a small proportion of the client's functionality requires the low-level API. For example:

```
import asyncio
from qth import Client
from qth_yarp import watch_event, send_event
from yarp import instantaneous_add

async def main():
    c = Client("mixed-qth-and-qth_yarp-client",
               "A client using both the qth and qth_yarp libraries.")

    # Regular Qth API
    def on_foo(topic, value):
        print("Event {} fired with value {}!".format(topic, value))
    await c.watch_event("example/foo", on_foo)

    # Also with qth_yarp (passing in the Qth Client)
    bar = watch_event("example/bar", qth_client=c)
    send_event("example/bar-plus-one",
               value=instantaneous_add(bar, 1),
               register=True,
               description="Incremented version of example/bar.",
               qth_client=c)

loop = asyncio.get_event_loop()
loop.create_task(main())
loop.run_forever()
```

The qth_yarp API is documented below:

CHAPTER 1

qth_yarp API

1.1 Qth to Value Functions

The following functions can be used to fetch a `yarp.Value` representing Qth properties or event.

qth_yarp.**get_property**(*topic*, *initial_value=None*, *register=False*, *description=None*, *one_to_many=False*, *delete_on_unregister=True*, *qth_client=None*, ***kwargs*)

Return a (continuous) `yarp.Value` containing the value of a qth property.

Parameters

topic [str] The Qth topic name. (Not verified for existance).

initial_value [object] The initial value to assign to the returned `yarp.Value` while waiting for the property to arrive from Qth. Defaults to `None` but you may wish to choose an alternative dummy value which won't crash later processing.

If (and only if) `register` is True, also sets the Qth property to this value.

register [bool] If True, registers this property with Qth. The 'description' argument must also be provided if this is True.

description [str] If `register` is True, the description of the property to include with the registration.

one_to_many [bool] If `register` is True, is this a one-to-many (True) or many-to-one (False) property. Defaults to many-to-one (False).

on_unregister [value] (Keyword-only argument). The value to set this property to when this Qth client disconnects. If not provided, no change is made. If this argument is used, set `delete_on_unregister` to False since it defaults to `True` and will conflict with this argument.

delete_on_unregister [bool] (Keyword-only argument). Should this value be deleted when this Qth client disconnects? Defaults to `True`.

qth_client [qth.Client] The Qth qth.Client object to use. If not provided, uses the client returned by *get_default_qth_client()*.

qth_yarp.**watch_event**(*topic*, *register=False*, *description=None*, *one_to_many=False*, *qth_client=None*, ***kwargs*)

Return an (instantaneous) yarp.Value representing a qth event.

Parameters

topic [str] The Qth topic name. (Not verified for existance).

register [bool] If True, registers this event with Qth. The 'description' argument must also be provided if this is True.

description [str] If register is True, the description of the event to include with the registration.

one_to_many [bool] If register is True, is this a one-to-many (True) or many-to-one (False) event. Defaults to many-to-one (False).

on_unregister [value] (Keyword-only argument). The value send for this event to when this Qth client disconnects. If not provided, event is sent.

qth_client [qth.Client] The Qth qth.Client object to use. If not provided, uses the client returned by *get_default_qth_client()*.

The next two functions use a yarp.Value to set or send a Qth property or event.

qth_yarp.**set_property**(*topic*, *value*, *register=False*, *description=None*, *one_to_many=True*, *delete_on_unregister=True*, *ignore_no_value=True*, *qth_client=None*, ***kwargs*)

Set a Qth property to the value of a continuous yarp.Value.

Parameters

topic [str] The Qth topic name. (Not verified for existance).

value [Value] The yarp.Value whose value will be written to the specified qth property.

register [bool] If True, registers this property with Qth. The 'description' argument must also be provided if this is True.

description [str] If register is True, the description of the property to include with the registration.

one_to_many [bool] If register is True, is this a one-to-many (True) or many-to-one (False) property. Defaults to one-to-many (True).

on_unregister [value] (Keyword-only argument). The value to set this property to when this Qth client disconnects. If not provided, no change is made. If this argument is used, set delete_on_unregister to False since it defaults to True and will conflict with this argument.

delete_on_unregister [bool] (Keyword-only argument). Should this value be deleted when this Qth client disconnects? Defaults to True.

ignore_no_value [bool] (Keyword-only argument). Should the property not be written when 'NoValue' is set. Defaults to True. If False, the property will be deleted when set to 'NoValue'.

qth_client [qth.Client] The Qth qth.Client object to use. If not provided, uses the client returned by *get_default_qth_client()*.

qth_yarp.**send_event**(*topic*, *value*, *register=False*, *description=None*, *one_to_many=True*, *qth_client=None*, ***kwargs*)

Return an (instantaneous) yarp.Value representing a qth event.

Parameters

topic [str] The Qth topic name. (Not verified for existance).

value [yarp.Value] An instantaneous yarp.Value whose changes will be turned into qth events.

register [bool] If True, registers this event with Qth. The 'description' argument must also be provided if this is True.

description [str] If register is True, the description of the event to include with the registration.

one_to_many [bool] If register is True, is this a one-to-many (True) or many-to-one (False) event. Defaults to one-to-many (True).

on_unregister [value] (Keyword-only argument). The value send for this event to when this Qth client disconnects. If not provided, event is sent.

qth_client [qth.Client] The Qth qth.Client object to use. If not provided, uses the client returned by *get_default_qth_client()*.

In all cases, the Values will not be updated, nor will value changes be sent to Qth until the asyncio mainloop is started. See *run_forever()*.

1.2 Specifying the Qth Client

The functions defined above accept a qth_client parameter giving the qth.Client object to use. The following function can be used (before any calls to the qth_yarp functions) to specify a specific qth.Client instance to use.

qth_yarp.**set_default_qth_client**(*client*)

Set the default qth.Client object to be used by qth_yarp.

If this function is not called, a Qth qth.Client will be created automatically. To fetch the Qth instance used, call:

qth_yarp.**get_default_qth_client**()

Get the default qth.Client object.

One will be created with the name qth_yarp_based_client if no client has been provided by set_default_client().

1.3 Main-loop utility function

As a convenience for simple scripts, the asyncio event loop can be run forever using the following function.

qth_yarp.**run_forever**()

Run the main event loop forever.

This function is a convenience method which is exactly equivalent to calling:

```
import asyncio
loop = asyncio.get_event_loop()
loop.run_forever()
```

Python Module Index

q

Index

G

Q

R

S

W