aioproperty

Release 0.3.2

Oct 28, 2019

Contents:

CHAPTER 1

Install

```
pip3 install aioproperty
```

CHAPTER 2

Documentation

You can find documentation here

CHAPTER 3

Description

aioproperty presents async properties with both async getter and setter in one place.

3.1 Example:

```
from aioproperty import aioproperty
import asyncio

class SomeClass:

    @aioproperty
    async def hello(self, value):
        await asyncio.sleep(1)
        return value

some_obj = SomeClass()
some_obj.hello = 'hello'

async def run():
    print(await some_obj.hello)

asyncio.run(run())
```

aioproperty is not a property in a classic meaning, it keeps values inside asincio tasks. Once you set a new value, it is scheduled in a task. If any task is running now, it will wait untill it is finished. When you get value, you actually get a current task, and you can await it to get a value. More of that: you can use math without awaiting like that:

```
other = some_obj.hello + ' byby'
print(await other)
```

We also introduce chaining:

```
class SomeClass:

    @aioproperty
    async def hello(self, value):
        await asyncio.sleep(1)
        return value

    @hello.chain
    async def some_more(self, value):
        push_value(value)
```

Chains works like a reducer. It applies each function to a new value iteratively. It can return value or return nothing. If nothing is return, the value will be kept asis. Any `chained` function is inserted by default in the end of chain, but you can control it with a priority parameter, or with is_first parameter. If is_first is True, it will be inserted in the beginning of the chain.

You can also use inheritance and modify inhereted properties with our special decorator `inject`:

```
class Parent:

    @aioproperty
    async def hello(self, value):
        await asyncio.sleep(1)
        return value

    @hello.chain
    async def some_more(self, value):
        push_value(value)

class Child(Parent):

    @inject(Parent.hello, priority=100)
    async def injected_method(self, value):
        print('hello from injection')

    #another form of injection using name of a property
    @inject('hello')
    async def injected_method_2(self, value):
        print('hello from injection 2')
```

Read more in our docs

CHAPTER 4

Indices and tables

- genindex
- modindex
- search