asyncbg Documentation

Release 0.12.0

Erik Moqvist

Apr 19, 2023

Contents

CHAPTER 1

Asyncio background tasks

Asyncio background tasks in Python 3.7 and later.

Run CPU intensive long running tasks without blocking the asyncio loop, implemented as a lightweight asyncio layer on top of the multiprocessing module.

Alternatively run tasks in other threads.

Project homepage: https://github.com/eerimoq/asyncbg

Documentation: https://asyncbg.readthedocs.org/en/latest

CHAPTER 2

Installation

```
pip install asyncbg
```

CHAPTER 3

Examples

There are more examples in the examples folder.

3.1 Call

Call `work(a, b)` in another process. The script output is `Result: 9`.

```
import asyncio
import asyncbg

def work(a, b):
    return a + b

async def main():
    result = await asyncbg.call(work, 4, 5)
    print(f'Result: {result}')

asyncio.run(main())
```

3.2 Process pool

Create a process pool with two workers, and call `work()` three times in it (up to two callbacks called in parallel).

```
import asyncio
import asyncbg

def work():
    pass

async def main():
```

(continues on next page)

(continued from previous page)

```
    with asyncbg.ProcessPoolExecutor(max_workers=2) as pool:
        await asyncio.gather(pool.call(work),
                             pool.call(work),
                             pool.call(work))

asyncio.run(main())
```

3.3 Call thread

Call `work(a, b)` in another thread. The script output is `Result:  9`.

```
import asyncio
import asyncbg

def work(a, b):
    return a + b

async def main():
    result = await asyncbg.call_thread(work, 4, 5)
    print(f'Result: {result}')

asyncio.run(main())
```

3.4 Thread pool

Create a thread pool with two workers, and call `work()` three times in it (up to two callbacks called in parallel).

```
import asyncio
import asyncbg

def work():
    pass

async def main():
    with asyncbg.ThreadPoolExecutor(max_workers=2) as pool:
        await asyncio.gather(pool.call(work),
                             pool.call(work),
                             pool.call(work))

asyncio.run(main())
```

CHAPTER 4

Functions and classes

asyncbg.**call**(*callback*, **args*, ***kwargs*)

Coroutine calling given callback with given arguments in another process.

Returns the value returned by the callback, or raises the exceptions raised by the callback.

Callback positional and keyword arguments can not be used for output, as the multiprocessing module does not support that.

Call `work()` in a worker process:

```
>>> def work():
>>>     pass
>>>
>>> asyncio.run(asyncbg.call(work))
```

class asyncbg.**ProcessPoolExecutor**(*max_workers=None*, *mp_context=None*, *initializer=None*, *initargs=()*)

Same as `concurrent.futures.ProcessPoolExecutor`, but with the `call()` method added.

Index

C

P