TypeError: An asyncio.Future, a coroutine or an awaitable is required

Asked 1 year, 4 months ago   Active 1 year, 4 months ago   Viewed 10k times

7

I'm trying to make an asynchronous web scraper using beautifulsoup and aiohttp.This is my initial code to start things.I'm getting a [TypeError: An asyncio.Future, a coroutine or an awaitable is required] and having a hard time figuring out what is wrong with my code.I am new to python and would appreciate any help regarding this.

```
import bs4
import asyncio
import aiohttp


async def parse(page):
    soup=bs4.BeautifulSoup(page,'html.parser')
    soup.prettify()
    print(soup.title)



async def request():
    async with aiohttp.ClientSession() as session:
        async with session.get("https://google.com") as resp:
            await parse(resp)



loop=asyncio.get_event_loop()
loop.run_until_complete(request)
```

Traceback:-

```
Traceback (most recent call last):
  File "C:\Users\User\Desktop\Bot\aio-req\parser.py", line 21, in <module>
    loop.run_until_complete(request)
  File "C:\Users\User\AppData\Local\Programs\Python\Python38-
32\lib\asyncio\base_events.py", line 591, in run_until_complete
    future = tasks.ensure_future(future, loop=self)
  File "C:\Users\User\AppData\Local\Programs\Python\Python38-32\lib\asyncio\tasks.py",
line 673, in ensure_future
    raise TypeError('An asyncio.Future, a coroutine or an awaitable is '
TypeError: An asyncio.Future, a coroutine or an awaitable is required
```

python   python-3.x   asynchronous   python-asyncio   aiohttp

Share Follow

edited Dec 25 '19 at 19:15
wwii
19.6k 5 32 69

asked Dec 25 '19 at 18:55
user7657046
154 1 1 9

2 When posting a question about code that produces an Exception, always include the complete Traceback - copy and paste it then format it as code (select it and type `ctrl-k`) – wwii Dec 25 '19 at 19:00

will keep that in mind.Added a traceback – user7657046 Dec 25 '19 at 19:03

alter line where request called as loop.run_until_complete(request()). you will encounter BeautifulSoup related issue when fixed but that is bs4 library related. it would be better to comment out entire parse function body and see if it works without any other dependency. – oetzi Dec 25 '19 at 19:27

Did you try refactoring like the client example in the aiohttp docs? – wwii Dec 25 '19 at 19:31

2 Answers

Active Oldest Votes

11

One issue is that `loop.run_until_complete(request)` should be `loop.run_until_complete(request())` - You actually have to call it for it to return a coroutine.

There are further problems - like you are passing an `aiohttp.ClientResponse` object to `parse` and treating it as text/html. I got it to work with the following but don't know if it fits your needs because `parse` is no longer a coroutine.

```
def parse(page):
    soup=bs4.BeautifulSoup(page,'html.parser')
    soup.prettify()
    return soup.title

async def fetch(session, url):
    async with session.get(url) as response:
        return await response.text()

async def request():
    async with aiohttp.ClientSession() as session:
        html = await fetch(session, "https://google.com")
        print(parse(html))

if __name__ == '__main__':
    loop=asyncio.get_event_loop()
    loop.run_until_complete(request())
```

This also works:

```
def parse(page):
    soup=bs4.BeautifulSoup(page,'html.parser')
    soup.prettify()
    print(soup.title)

async def request():
    async with aiohttp.ClientSession() as session:
        async with session.get("https://google.com") as resp:
            parse(await resp.text())
```

And finally, your original code, passing an awaitable response object to `parse` then awaiting for `page.text()` .

```
async def parse(page):
    soup=bs4.BeautifulSoup(await page.text(),'html.parser')
    soup.prettify()
    print(soup.title)

async def request():
    async with aiohttp.ClientSession() as session:
        async with session.get("https://google.com") as resp:
            await parse(resp)
```

Share Follow

edited Dec 25 '19 at 20:28

answered Dec 25 '19 at 19:43

wwii
19.6k 5 32 69

correction:: `aiohttp.ClientResponse` not `aiohttp.response` – wwii Dec 25 '19 at 19:56

Yeah i changed my code and it works now.I would not have noticed not placing the brackets after request if you did not point that out.Thanks again big time! – user7657046 Dec 25 '19 at 19:59

@user7657046 There is a caveat in the Client Quickstart page about making a new ClientSession object for each request. You may want to factor that into your app. – wwii Dec 25 '19 at 20:00

I will incorporate that into my code.Thanks for your suggestion,once again. – user7657046 Dec 25 '19 at 20:17

2

I changed my code to this and it works now.

```
import bs4
import asyncio
import aiohttp


async def parse(page):
    soup=bs4.BeautifulSoup(page,'html.parser')
    soup.prettify()
    print(soup.title)



async def request():
    async with aiohttp.ClientSession() as session:
        async with session.get("https://google.com") as resp:
            html=await resp.text()
            await parse(html)



loop=asyncio.get_event_loop()
loop.run_until_complete(request())
```

user7657046
154 1 1 9