Context var

Контекстные переменные имеют свое значение для разных мест вызова приложения.

Они изначально поддерживают использование библиотеки asyncio.

документация

Пример. Работа с обычной глобальной переменной:

```
import asyncio


global_var = None


async def printer(i: int):
    global global_var
    print(f"printer {i} global_var={global_var}")
    global_var = i
    print(f"printer {i} global_var={global_var}")


async def main():
    await asyncio.gather(*[printer(i + 1) for i in range(3)])


asyncio.run(main())
```

```
printer 1 global_var=None
printer 1 global_var=1
printer 2 global_var=1
printer 2 global_var=2
printer 3 global_var=2
printer 3 global_var=3
printer 4 global_var=3
```

А теперь тот же код перепишем на **context var:**

```
import asyncio
from contextvars import ContextVar

global_var: ContextVar[int] = ContextVar('global_var', default=None)


async def printer(i: int):
    print(f"printer {i} global_var={global_var.get()}")
    global_var.set(i)
    print(f"printer {i} global_var={global_var.get()}")


async def main():
    await asyncio.gather(*[printer(i + 1) for i in range(3)])


asyncio.run(main())
```

```
printer 1 global_var=None
printer 1 global_var=1
printer 2 global_var=None
printer 2 global_var=2
printer 3 global_var=None
printer 3 global_var=3
```

Пример использования. У нас HTTP-сервер. На каждый запрос мы ходим в одну или несколько баз, делаем запросы в сеть. Все эти операции мы логируем и сохраняем. Сервер у нас асинхронный, и в разные моменты времени исполняются операции из разных запросов. Мы хотим помечать при логировании, к какому конкретному запросу относится та или иная строчка служебной информации.

Благодаря **ContextVar** мы можем каждому запросу присваивать уникальный идентификатор и дописывать его к каждой записи логов. Теперь при необходимости мы сможем любым известным нам способом отфильтровать «простыню» из информации и провести свои исследования в подозрительных случаях.