Asyncio и конкурентное программирование на Python

Мэттью Фаулер

MANNING

ДМК ПРЕСС

Оглавление

1 Первое знакомство с asyncio 21
2 Основы asyncio 45
3 Первое приложение asyncio 74
4 Конкурентные веб-запросы 101
5 Неблокирующие драйверы баз данных 130
6 Счетные задачи 157
7 Решение проблем блокирования с помощью потоков 189
8 Потоки данных 223
9 Веб-приложения 251
10 Микросервисы 279
11 Синхронизация 303
12 Асинхронные очереди 327
13 Управление подпроцессами 350
14 Продвинутое использование asyncio 365

Содержание

Оглавление 6
Предисловие 12
Благодарности 14
Об этой книге 15
Об авторе 19
Об иллюстрации на обложке 20

1 *Первое знакомство с asyncio* 21
1.1 Что такое asyncio? 22
1.2 Что такое ограниченность производительностью ввода-вывода и ограниченность быстродействием процессора 24
1.3 Конкурентность, параллелизм и многозадачность 25
1.3.1 Конкурентность 25
1.3.2 Параллелизм 26
1.3.3 Различие между конкурентностью и параллелизмом 27
1.3.4 Что такое многозадачность 28
1.3.5 Преимущества кооперативной многозадачности 28
1.4 Процессы, потоки, многопоточность и многопроцессность 29
1.4.1 Процесс 29
1.4.2 Поток 29
1.5 Глобальная блокировка интерпретатора 33
1.5.1 Освобождается ли когда-нибудь GIL? 37
1.5.2 Asyncio и GIL 39
1.6 Как работает однопоточная конкурентность 39
1.6.1 Что такое сокет? 39
1.7 Как работает цикл событий 41
Резюме 44

2 *Основы asyncio* 45
2.1 Знакомство с сопрограммами 46
2.1.1 Создание сопрограмм с помощью ключевого слова async 46
2.1.2 Приостановка выполнения с помощью ключевого слова await 48
2.2 Моделирование длительных операций с помощью sleep 49
2.3 Конкурентное выполнение с помощью задач 52

2.3.1 Основы создания задач ... 52
2.3.2 Конкурентное выполнение нескольких задач ... 53
2.4 Снятие задач и задание тайм-аутов ... 56
2.4.1 Снятие задач ... 56
2.4.2 Задание тайм-аута и снятие с помощью wait_for ... 57
2.5 Задачи, сопрограммы, будущие объекты и объекты, допускающие ожидание ... 59
2.5.1 Введение в будущие объекты ... 59
2.5.2 Связь между будущими объектами, задачами и сопрограммами ... 61
2.6 Измерение времени выполнения сопрограммы с помощью декораторов ... 62
2.7 Ловушки сопрограмм и задач ... 65
2.7.1 Выполнение счетного кода ... 65
2.7.2 Выполнение блокирующих API ... 67
2.8 Ручное управление циклом событий ... 68
2.8.1 Создание цикла событий вручную ... 69
2.8.2 Получение доступа к циклу событий ... 69
2.9 Отладочный режим ... 70
2.9.1 Использование asyncio.run ... 70
2.9.2 Использование аргументов командной строки ... 71
2.9.3 Использование переменных окружения ... 71
Резюме ... 72

3 *Первое приложение asyncio* ... 74
3.1 Работа с блокирующими сокетами ... 75
3.2 Подключение к серверу с помощью telnet ... 78
3.2.1 Чтение данных из сокета и запись данных в сокет ... 79
3.2.2 Разрешение нескольких подключений и опасности блокирования ... 80
3.3 Работа с неблокирующими сокетами ... 82
3.4 Использование модуля selectors для построения цикла событий сокетов ... 86
3.5 Эхо-сервер средствами цикла событий asyncio ... 89
3.5.1 Сопрограммы цикла событий для сокетов ... 89
3.5.2 Проектирование асинхронного эхо-сервера ... 90
3.5.3 Обработка ошибок в задачах ... 92
3.6 Корректная остановка ... 94
3.6.1 Прослушивание сигналов ... 95
3.6.2 Ожидание завершения начатых задач ... 96
Резюме ... 99

4 *Конкурентные веб-запросы* ... 101
4.1 Введение в aiohttp ... 102
4.2 Асинхронные контекстные менеджеры ... 103
4.2.1 Отправка веб-запроса с помощью aiohttp ... 105
4.2.2 Задание тайм-аутов в aiohttp ... 107
4.3 И снова о конкурентном выполнении задач ... 108
4.4 Конкурентное выполнение запросов с помощью gather ... 111
4.4.1 Обработка исключений при использовании gather ... 113
4.5 Обработка результатов по мере поступления ... 115
4.5.1 Тайм-ауты в сочетании с as_completed ... 117

4.6 Точный контроль с помощью wait 119
4.6.1 Ожидание завершения всех задач 119
4.6.2 Наблюдение за исключениями 122
4.6.3 Обработка результатов по мере завершения 123
4.6.4 Обработка тайм-аутов 126
4.6.5 Зачем оборачивать сопрограммы задачами? 127
Резюме 128

5 *Неблокирующие драйверы баз данных* 130
5.1 Введение в asyncpg 131
5.2 Подключение к базе данных Postgres 131
5.3 Определение схемы базы данных 133
5.4 Выполнение запросов с помощью asyncpg 135
5.5 Конкурентное выполнение запросов с помощью пулов подключений 138
5.5.1 Вставка случайных SKU в базу данных о товарах 138
5.5.2 Создание пула подключений для конкурентного выполнения запросов 142
5.6 Управление транзакциями в asyncpg 146
5.6.1 Вложенные транзакции 148
5.6.2 Ручное управление транзакциями 149
5.7 Асинхронные генераторы и потоковая обработка результирующих наборов 151
5.7.1 Введение в асинхронные генераторы 151
5.7.2 Использование асинхронных генераторов и потокового курсора 153
Резюме 156

6 *Счетные задачи* 157
6.1 Введение в библиотеку multiprocessing 158
6.2 Использование пулов процессов 160
6.2.1 Асинхронное получение результатов 161
6.3 Использование исполнителей пула процессов в сочетании с asyncio 162
6.3.1 Введение в исполнители пула процессов 162
6.3.2 Исполнители пула процессов в сочетании с циклом событий 164
6.4 Решение задачи с помощью MapReduce и asyncio 166
6.4.1 Простой пример MapReduce 167
6.4.2 Набор данных Google Books Ngram 169
6.4.3 Применение asyncio для отображения и редукции 170
6.5 Разделяемые данные и блокировки 175
6.5.1 Разделение данных и состояние гонки 176
6.5.2 Синхронизация с помощью блокировок 179
6.5.3 Разделение данных в пулах процессов 181
6.6 Несколько процессов и несколько циклов событий 184
Резюме 188

7 *Решение проблем блокирования с помощью потоков* 189
7.1 Введение в модуль threading 190

7.2 Совместное использование потоков и asyncio 194
7.2.1 Введение в библиотеку requests 194
7.2.2 Знакомство с исполнителями пула потоков 195
7.2.3 Исполнители пула потоков и asyncio 197
7.2.4 Исполнители по умолчанию 198
7.3 Блокировки, разделяемые данные и взаимоблокировки 200
7.3.1 Реентерабельные блокировки 201
7.3.2 Взаимоблокировки 204
7.4 Циклы событий в отдельных потоках 206
7.4.1 Введение в Tkinter 207
7.4.2 Построение отзывчивого UI с помощью asyncio и потоков 209
7.5 Использование потоков для выполнения счетных задач 217
7.5.1 hashlib и многопоточность 217
7.5.2 Многопоточность и NumPy 220
Резюме 222

8 *Потоки данных* 223
8.1 Введение в потоки данных 224
8.2 Транспортные механизмы и протоколы 224
8.3 Потоковые читатели и писатели 228
8.4 Неблокирующий ввод данных из командной строки 231
8.4.1 Режим терминала без обработки и сопрограмма read 235
8.5 Создание серверов 242
8.6 Создание чат-сервера и его клиента 244
Резюме 249

9 *Веб-приложения* 251
9.1 Разработка REST API с помощью aiohttp 252
9.1.1 Что такое REST? 252
9.1.2 Основы разработки серверов на базе aiohttp 253
9.1.3 Подключение к базе данных и получение результатов 255
9.1.4 Сравнение aiohttp и Flask 260
9.2 Асинхронный интерфейс серверного шлюза 263
9.2.1 Сравнение ASGI и WSGI 263
9.3 Реализация ASGI в Starlette 264
9.3.1 Оконечная REST-точка в Starlette 265
9.3.2 WebSockets и Starlette 266
9.4 Асинхронные представления Django 269
9.4.1 Выполнение блокирующих работ в асинхронном представлении 275
9.4.2 Использование асинхронного кода в синхронных представлениях 277
Резюме 278

10 *Микросервисы* 279
10.1 Зачем нужны микросервисы? 280
10.1.1 Сложность кода 281
10.1.2 Масштабируемость 281
10.1.3 Независимость от команды и технологического стека 281
10.1.4 Чем может помочь asyncio? 281
10.2 Введение в паттерн backend-for-frontend 282
10.3 Реализация API списка товаров 283

10.3.1 Сервис избранного 284
10.3.2 Реализация базовых сервисов 284
10.3.3 Реализация сервиса backend-for-frontend 289
10.3.4 Повтор неудачных запросов 294
10.3.5 Паттерн Прерыватель 297
Резюме 302

11 *Синхронизация* 303
11.1 Природа ошибок в модели однопоточной конкурентности 304
11.2 Блокировки 309
11.3 Ограничение уровня конкурентности с помощью семафоров 312
11.3.1 Ограниченные семафоры 315
11.4 Уведомление задач с помощью событий 317
11.5 Условия 322
Резюме 326

12 *Асинхронные очереди* 327
12.1 Основы асинхронных очередей 328
12.1.1 Очереди в веб-приложениях 335
12.1.2 Очередь в веб-роботе 338
12.2 Очереди с приоритетами 341
12.3 LIFO-очереди 347
Резюме 349

13 *Управление подпроцессами* 350
13.1 Создание подпроцесса 351
13.1.1 Управление стандартным выводом 353
13.1.2 Конкурентное выполнение подпроцессов 357
13.2 Взаимодействие с подпроцессами 360
Резюме 363

14 *Продвинутое использование asyncio* 365
14.1 API, допускающие сопрограммы и функции 366
14.2 Контекстные переменные 368
14.3 Принудительный запуск итерации цикла событий 370
14.4 Использование других реализаций цикла событий 371
14.5 Создание собственного цикла событий 373
14.5.1 Сопрограммы и генераторы 373
14.5.2 Использовать сопрограммы на основе генераторов не рекомендуется 374
14.5.3 Нестандартные объекты, допускающие ожидание 376
14.5.4 Сокеты и будущие объекты 378
14.5.5 Реализация задачи 381
14.5.6 Реализация цикла событий 382
14.5.7 Реализация сервера с использованием своего цикла событий 385
Резюме 387

Предметный указатель 388