Краткий обзор основных возможностей

Всем привет! В прошлом модуле мы познакомились с тем, как реализовано асинхронное программирование в Python. Но какую проблему прежде всего решали авторы языка/библиотеки Asyncio? Всё дело в том, что Python широко известен как невероятно медленный язык. Да, наш любимый язык, согласно разным бенчмаркам, медленнее PHP или JS. Но действительно ли всё так плохо?

С одной стороны, да, наверно, если мы хотим заниматься дроблением чисел, то лучше или выбрать специальные библиотеки, где ограничения Python, вроде GIL, преодолены, или другой язык. Но если мы говорим об IO-bound-задачах, то тут всё не так уж плохо. И по счастливому совпадению весь веб строится не на CPU-bound-задачах, а именно на IO. И тут показывает всю свою мощь асинхронное программирование. Ну подумайте сами: input/output — это и есть всё, что касается сети. Сделать запрос на handshake — output, получить подтверждение — input. Зачем нам блокироваться, ожидая ответа от клиента с его не особо хорошим соединением, когда мы могли бы в это время начать работу с другим клиентом, у которого соединение тоже не особо быстрое, если сравнивать со временем работы CPU.

Райан Даль, создатель Node.js, мотивируя создание фреймворка — а это, к слову, асинхронный серверный JS, — привёл такую аналогию. Если мы попробуем измерять в тактах CPU время, затраченное на выполнение того или иного действия, то получим такую таблицу:

Устройство	Такты CPU	~ Человеческий аналог
L1 cache	3	1 секунда
L2 cache	14	4,6 секунды
RAM	250	83 секунды
Disk	41000000	158 дней
Network	240000000	2,5 года

В первой колонке — устройство, к которому мы обращаемся. Первые два, cache L1 и L2, — это специальные устройства памяти, которые находятся прямо в самом процессоре. Доступ к ним очень быстрый, но и объём памяти там небольшой. Во второй колонке находится примерное количество тактов CPU. Один CPU-такт — это время, нужное процессору на проведение некой базовой операции, например сложение двух чисел. В третьей — примерная аналогия в человеческом времени.

Как вы понимаете, асинхронное программирование и веб просто созданы друг для друга. И сегодня мы с вами разберём один из самых популярных асинхронных фреймворков. Называется он FastAPI.

Кратко пробежимся по его ключевым особенностям.

Прежде всего, он асинхронный. Когда разрабатывались фреймворки Django или Flask, в мире Python ещё не было такого понятия, как асинхронность (во всяком случае, это было не распространено). Но не пугайтесь, писать асинхронный код на нём совсем несложно, нужно не забывать только пару основных правил, которые мы и так уже знаем: как использовать async/await и как работать с синхронным кодом.
А раз он асинхронный, значит, быстрый. Некоторые бенчмарки показывают сопоставимую скорость работы с Node.js и GO.

Добавьте сюда простоту написания кода эндпоинтов — и мы получаем очень популярный фреймворк, которым пользуются бигтех-компании вроде Google и Microsoft. А это означает, что есть огромное комьюнити и все вопросы уже давно разжеваны в интернете. 37 тысяч звёздочек на GitHub как-никак.

Кроме того, этот фреймворк аннотирован на 100%. Кажется, что это мелочь, но как бы не так. Прежде всего это даёт отличную поддержку в IDE. Автодополнение работает великолепно. Но это не всё: многие фичи, вроде валидации параметров, работают за счёт аннотации типов. В итоге код начинает выглядеть лаконично.

Вкусовщина, но он довольно-таки интуитивен. Чтобы разобраться в нём, хватит вечерочка-двух с периодическим обращением к документации.

Его достаточно просто деплоить. Можно обойтись без настройки nginx и WSGI, а использовать фреймворк, который предлагает библиотека, — uvicorn. Это асинхронный протокол, наследник WSGI.

Очередное следствие аннотации типов: в FastAPI из коробки работает автогенерация документации в стандарте OpenAPI — мы с вами его уже разбирали, и ReDoc. Это делает фреймворк уже готовым к деплойменту в продакшен без дополнительных усилий по описанию эндпоинтов. Свежая документация всегда под рукой.

Ну и наконец, last, but not least: у этого фреймворка потрясающая документация. Один минус: перевода на русский пока нет. Если вы до сих пор боялись читать документацию, то это отличный шанс начать. Понятная, структурированная, с юмором и повествующая от простого к сложному. Кстати, можете побыть контрибьюторами в переводе этой документации на русский язык. Полезно и для практики языка, и для запоминания особенностей фреймворка. Ссылку смотрите под видео.

Итак, мы узнали, почему асинхронность так хорошо подходит для написания веб-приложений, и в общих чертах познакомились с ключевыми фичами самого популярного асинхронного фреймворка на Python — FastAPI. В следующем уроке мы познакомимся с ним на практике.