Asynchronous Programming and Web Framework

3월에 비동기 프로그래밍이라는 주제로 발표를 하였는데.. 이어서 하겠습니다..

기억이 가물가물하기 때문에 전 내용을 간단하게 짚고 넘어갑시다!

Asynchronous Programming

비동기 프로그래밍?

- 결과를 기다리지 않고 다음 코드를 실행하는 흐름을 대표하는 개념
- 자원을 더 효율적으로 사용
- 동시성과 Non-Blocking I/O 와 독립된 정의로 다루어야 한다

비동기 프로그래밍에서 결과를 처리하는 방식

- 함수 전달을 통해 처리(Callback)

- 언어에서 지원하는 방식(Future, Promise, Python 코루틴, Go 고루틴 등)

Asynchronous Wait (Await)

- 비동기 프로그래밍으로 실행 한 작업과 원래 실행 흐름을 묶어주는 중요한 기법

- Await 를 통해 실행 흐름을 제어

- 추상화의 수준을 절차 중심 프로그래밍 언어의 수준으로 낮추는 주된 요인

Asynchronous Programming and Web Framework

비동기 프로그래밍을 사용하지 않는 웹 프레임워크

- 요청을 받아 처리를 한 후 결과를 되돌려 주는 작업을 하나의 일꾼이 수행

- 많은 요청을 처리하기 위해 여러 일꾼을 이용 -> 물리적인 제약이 생김

어떻게 비동기 프로그래밍을 적용?

- 요청을 처리 로직과 연결해주는 일에 드는 자원 < 처리 로직에서 사용하는 자원

- 요청을 처리 로직에 연결하는 일과 로직의 일 사이의 관계를 때어냄

- 요청과 처리 로직을 연결하는 과정에서 비동기 프로그래밍 방식을 사용

Thread Pool

- 미리 만들어둔 스레드를 풀에서 필요할 때 하나씩 꺼내어 사용
- 많은 곳에서 스레드 풀을 구현체로 사용
- 동시성과 병렬성을 확보

문제점 & 해결 방안

- 오래 걸리는 작업이 많아지는 경우
- 스레드가 고갈 되지 않게 신경써야됨

- 오래 걸리는 작업을 위한 별도의 스레드 풀을 사용

- 비동기 작업 내부에 일어나는 I/O 를 Non-Blocking I/O 로 처리