**Redis란?**

* 비정형 데이터(Key, Value 구조)를 저장하고 관리하기 위한 비관계형 데이터 베이스 관리 시스템
* Key, Value 구조 -> 쿼리를 사용할 필요가 없다!
* 데이터를 디스크에 쓰는 구조(영속적)가 아니라 인메모리 방식 -> 속도 빠름
* **Single Threaded** -> 한 번에 하나의 명령만! -> 그러나 get, set 명령어의 경우 초당 10만 개 이상 처리할 수 있을 만큼 빠름! 시간에 대해 크게 걱정하지 않아도 OK ->분산서버에서 동시성 문제가 발생하지 않음 (분산 서버에 적합!)
* String, Lists, Sets, Sorted Sets, Hashes 자료 구조를 지원합니다.
  + **String** : 일반적인 key - value 구조의 형태. (ex. 챗봇 프로젝트에서 쓸 예정)
  + **Sets** : String의 집합(중복X)입니다. 여러 개의 값을 하나의 value에 넣을 수 있다. (ex. 글 관련 태그)
  + **Sorted Sets** : Set 구조에 정렬 Sort를 적용한 구조로 랭킹 보드 서버 같은 구현에 사용할 수 있습니다. (ex. 게임 랭킹)
  + **Lists** : Double Linked List 데이터 구조 -> 따라서 처음 마지막의 crd는 빠르지만, 중간의 crd는 느림

**레디스 사용 이유**

* 캐시 용도
  + 대용량 트래픽 시, 데이터 베이스가 과부하 될 수 있기 때문에 레디스를 도입해서 사용
    - 락을 사용해야 할 경우 api가 들어올때마다 발생하기 때문에, 데이터 베이스까지 사용한다면 시간이 많이 걸릴 것이다. 그리고 lock으로 이용하기 위해 사용하는 기능이므로 영속화 할 필요가 없기에 상대적으로 시간이 빠른 redis를 사용하는게 적합해보인다.
  + 인메모리 방식이기 때문에 ttl 설정으로 공간 낭비를 막아주는 기능이 있다
    - lock목적으로 redis를 사용할 예정이라면, 만약에 서버가 다운되더라도 ttl을 동해 lock기능을 정상화 할 수 있으므로, redis를 사용하는 것이 바람직해 보인다.
* 분산 처리
  + 레디스의 경우 단일 쓰레드를 이용하기 때문에, 동시성 문제가 없다. 따라서 분산 서버일 경우 유용하다.
    - 규모가 큰 프로젝트라면 추후 서버가 늘어나는 걸 대비해서 redis를 이용하는게 좋아보인다.

**레디스 캐시 서버 패턴 2가지**

* **Look aside cache 패턴 (락으로 redis를 사용한다면 요 패턴!)**
  + 1. 클라이언트가 데이터를 요청
  + 2. 웹서버는 데이터가 존재하는지 Cache 서버에 먼저 확인
  + 3. Cache 서버에 데이터가 있으면 DB에 데이터를 조회하지 않고 Cache 서버에 있는 결과값을 클라이언트에게 바로 반환 (Cache Hit)
  + 4. Cache 서버에 데이터가 없으면 DB에 데이터를 조회하여 Cache 서버에 저장하고 결과값을 클라이언트에게 반환 (Cache Miss)
* **Write Back 패턴**
  + 1. 웹서버는 모든 데이터를 Cache 서버에 저장
  + 2. Cache 서버에 특정 시간 동안 데이터가 저장됨
  + 3. Cache 서버에 있는 데이터를 DB에 저장
  + 4. DB에 저장된 Cache 서버의 데이터를 삭제