



스프링 스터디 2주차 (2021.08.22)

- 섹션 3. 회원 관리 예제 - 백엔드 개발
- 섹션 4. 스프링 빈과 의존 관계
- 섹션 5. 회원 관리 예제 - 웹 MVC 개발



회원 관리의 웹 MVC 구조를 고려하여 백엔드 개발과 테스트 코드 작성을 진행해 보았다!

스프링 빈과 의존 관계에 대해 공부하고, 컴포넌트 스캔 방식과, 직접 스프링 빈을 등록하는 방식 둘 다 적용해 보았다!

Spring의 동시성 문제?

Repository class에서 메모리를 저장하기 위해 Map과 long .. 을 활용한다.

그런데 이 때 공유되는 변수는 **동시성 문제**를 고려해야 한다는 언급이 나온다.



실무에서는 동시성 문제가 있을 수 있어서 공유되는 변수일 때는~

HashMap -> ConcurrentHashMap

Long -> AtomicLong

여기서는 예제니까 단순하게 하겠습니다.. ㅎㅎ

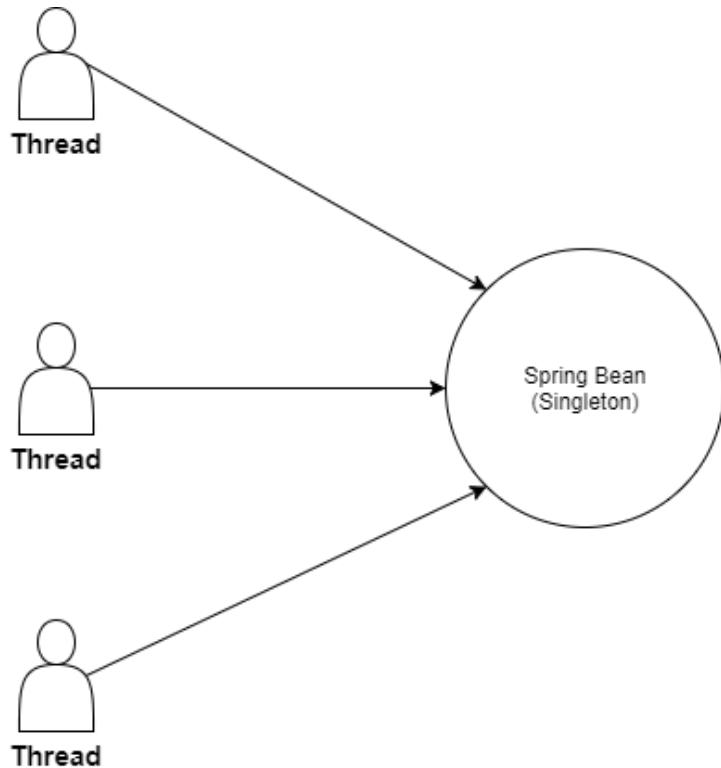
동시성 문제가 무엇인지, **ConcurrentHashMap**과 **AtomicLong** 그리고 다른 방법은 무엇이 있는지 알아보자!

동시성 문제

동시성 문제의 원인!



Multi-threading 환경에서 발생!



기본적으로 Spring MVC에서 사용자로부터 요청을 받게 되면, Container에서 **Thread**를 생성하고, 이에 대한 처리를 한 뒤 응답을 주는 구조이기 때문에, 기본적으로 **멀티 스레드** 환경이라고 생각을 해야 한다.

자바에서 멀티 스레드를 잘 사용하면 좋은 성능을 내는 프로그램을 개발할 수 있지만, 스레드 간 동기화(Synchronize) 문제를 필수적으로 해결해야 한다.

ex) Tomcat 같은 경우 사용자의 요청을 병렬적으로 처리하기 위해 멀티 쓰레드를 활용하고 있고, 요청당 쓰레드 1개의 할당이 이루어진다.

동시성(Concurrency)?

사전적 의미는 하나의 CPU 코어에서 시간 분할(Time sharing)을 통하여 여러 일을 처리하는 것 처럼 보여지게 하는 기법을 의미!

대중적으로는 **여러 요청이 동시에 동일한 자원(data)에 접근하고 수정**하려는 것을 의미!

- 동시성 고려를 안해준다면....



초콜릿을 아껴 먹으려고 10개를 서랍에 넣어 두었다. 하지만 내가 모르는 사이에 동료들이 몰래 훔쳐 먹었다. 나는 10개가 남은 것을 기대하고 서랍을 열었지만 빈 상자만...ㅠㅠ

데이터를 수정하여 저장했지만 다시 조회했을 때 다른 값이 반환 되는 경우가 생긴다!

쇼핑몰에서 상품의 재고 수량이나 경매 시스템에서 실시간 호가를 한 곳에서 관리를 한다면 동시성을 고려해야 한다! 상품이 더 팔리거나 항상 커져야 할 경매 호가가 낮아질 수도 있기 때문이다!

동시성 문제 해결 방법 (Thread-safe!)

Synchronized 키워드 사용

Java에서는 thread-safe를 지원하는 synchronized 키워드를 사용하여 스레드 간 동기화를 시켜준다.

synchronized 키워드를 통해 베타 동기를 제공하는데, 여러 개의 스레드가 하나의 공유 자원에 접근하게 될 때, 현재 데이터를 사용하고 있는 스레드를 제외하고 나머지 스레드들은 데이터에 접근할 수 없게 된다. 공유 자원을 사용하는 스레드가 존재하면, 베타 동기에서 대기하게 한다.

하지만 synchronized는 마치 줄을 세워 처리하는 것과 같으므로 무분별하게 사용할 경우 프로그램 성능이 떨어지게 된다. synchronized의 특징은 불공정 방법(unfair)이라 해서 스레드 간 락(lock)을 획득하는 순서를 보장해 주지 않는다.

Lock

스레드 동기화 매커니즘 중에 java.util.concurrent에 존재하는 Lock 이 synchronized와 유사하다. 그 중에 ReentrantLock은 공정 방법과 불공정 방법으로 설정을 할 수 있다.

만약 스레드가 동시에 들어오는 경우가 4개 이상인 경우 혹은, 락을 모니터링 하는 경우, 동기화 코드가 복잡한 경우에는 효율이 더 좋다.

wait(), notify()

synchronized를 이용해서 동기화하는 경우 한 스레드가 Lock을 너무 오래 가지게 되면 다른 스레드는 아무것도 못하고 기다리는 상황이 발생한다. 이 걸 방지하기 위해 나온게 **wait()**, **notify()** 메서드

이다.

wait()를 호출하면 스레드는 락을 반납하고 기다리게 한다. 그러면 다른 스레드가 Lock을 얻어 작업을 수행하게 된다.

notify()를 호출하면 **wait()** 호출로 인해 Lock을 반납한 스레드가 다시 Lock을 얻어 작업을 수행 할 수 있다.



But... 과도한 동기화는 피해야 한다!

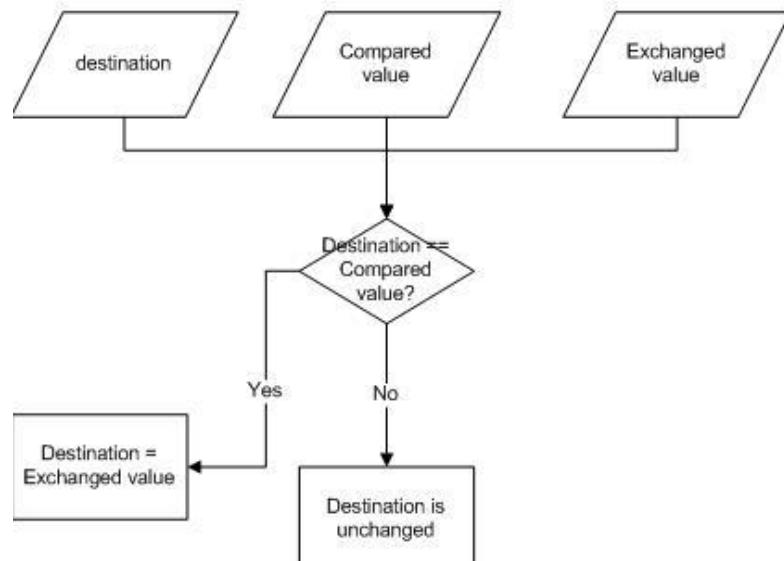
ConcurrentHashMap, AtomicLong...

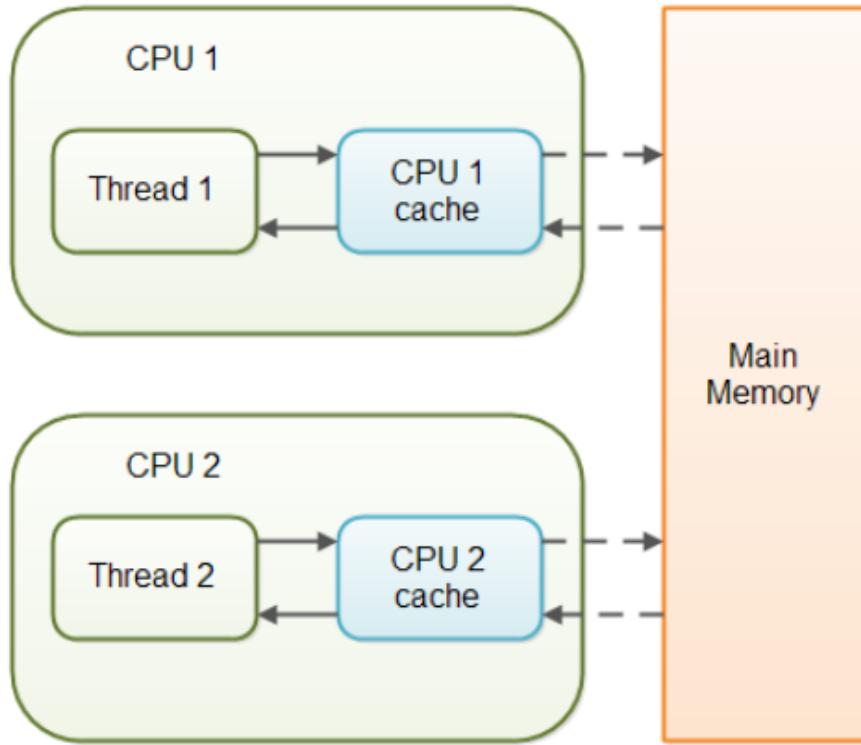
위의 방법들 보다는 고수준 동시성 유ти리티를 애용하자!

CAS(Compare And Swap) 알고리즘을 적용한 **volatile** 키워드를 활용해 성능이 더 좋다.

CAS

Compare And Swap





현재 쓰레드에 저장된 값을 CPU 캐시에 저장된 값과 비교하지 않고 메인 메모리에 직접 접근해 값을 비교하여 일치하는 경우 새로운 값으로 교체하고, 일치하지 않는 경우 실패하고 재 시도를 한다. 하지만 CAS는 ABA 문제가 존재한다. 또 이를 해결하기 위한 게 Double Check CAS가 있다.

volatile 키워드의 변수를 사용한 Atomic 클래스

자바에서는 CAS 알고리즘을 사용하는 Atomic 클래스를 제공하는데, 내부적으로 volatile 키워드의 변수를 사용하여 적용한다.

ConcurrentHashMap

```
private final Map<Long, Member> store = new ConcurrentHashMap<>();
...
store.putIfAbsent(member.getId(), member);
store.getOrDefault(id);
```

AtomicLong

```
private final AtomicLong sequence = new AtomicLong(1);
...

```

```
sequence(sequence.getAndIncrement());
```

DB에서의 해결

비관적인 방법

현재 수정하려는 데이터에 Lock을 거는 방식

낙관적인 방법

수정하려는 데이터는 나만 수정할 것이라는 낙관적인 생각의 방법

테이블에 version이라는 숫자컬럼 또는 updated_at이라는 시간컬럼을 만들어서 수정될 때마다 1씩 증가하거나, 현재 시간으로 갱신

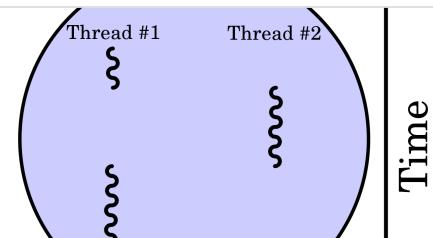
값을 수정할 때 version이 동일하면 수정이 가능해지고, 동일하지 않으면 수정에 실패

Ref.

멀티스레딩 - 위키백과, 우리 모두의 백과사전

멀티스레딩(multithreading) 컴퓨터는 여러 개의 스레드를 효과적으로 실행할 수 있는 하드웨어 지원을 갖추고 있다. 이는 스레드가 모두 같은 주소 공간에서 동작하여 하나의 CPU 캐시 공유 집합과 하나의 변환 버퍼(TLB)만 있는

W <https://ko.wikipedia.org/wiki/%EB%A9%80%ED%8B%B0%EC%8A%A4%EB%A0%88%EB%94%A9>



Spring MVC - Spring MVC 동기화와 JPA 잠금기법

Spring MVC 동시성 테스트 이번 포스팅에서는 동기화 관련 문제를 다루어 보도록 하겠습니다. 우선, 동기화에 대해 알아보고, Spring MVC를 이용해 발생 할 수 있는 문제점을 보도록 하겠습니다. 마지막으로, 이를 해결하는 방법도 알

₩ <https://galid1.tistory.com/790>

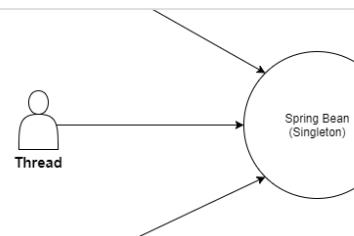
```
t 111 : -3  
t 111 : -2  
t 111 : -1  
t 111 : 0  
t 111 : 1  
t 111 : 2
```

```
BUILD SUCCESSFUL in 9s  
3 actionable tasks: 1 executed, 2 up-
```

[Java] synchronized: multi-thread 환경에서 동기화 문제를 해결하는 키워드

자바에서 멀티 스레드를 잘 사용하면 좋은 성능을 내는 프로그램을 개발할 수 있지만, 스레드 간 동기화 문제를 필수적으로 해결해야 합니다. 예를 들어 스레드 간 공유 자원 으로 사용하고 있는 데이터가 있을 경우, 여러 개의 스레드가 하나의 데이터에 접근할

₩ <https://ooeunz.tistory.com/110>



[DB] 동시성 문제 해결방법

예를 들어 초콜릿을 아껴 먹으려고 10개를 서랍에 넣어두었습니다. 하지만 내가 모르는 사이에 동료들이 몰래 훔쳐먹었습니다. 나는 10개가 남은 것을 기대하고 서랍을 열었지만 빈 상자만 남았습니다. 이와 같은 현상은 DB에서 더욱

<https://chrisjune-13837.medium.com/db-%EB%8F%99%EC%8B%9C%EC%84%B1-%EB%AC%B8%EC%A0%9C-%ED%95%B4%EA%B2%B0%EB%B0%A9%EB%B2%95-f5e52e2e3>

Concurrency

자바 동시성을 활용한 예약 시스템 구현

개발을 하다보면 예약 프로그램 구현이 필요한 경우가 있습니다. 예를 들면 시설물 사용 신청, 단체 예약, 영화 예약 등이 있는데.. 예약 시스템의 특징은 여러 사람이 동시에 신청할 가능성이 있다는 것입니다. 이 부분이 핵심인데, 최악

⌚ <https://webdevtechblog.com/%EC%9E%90%EB%B0%94-%EB%8F%99%EC%8B%9C%EC%84%B1%EC%9D%84-%ED%99%9C%EC%9A%A9%ED%95%9C-%EC%98%88%EC%95%BD-%EC%8B%9C%EC%8A%A4%ED%85%9C-%EA%B5%AC%ED%98%84-ad39f762>

스프링에서 동시성 문제 해결하기

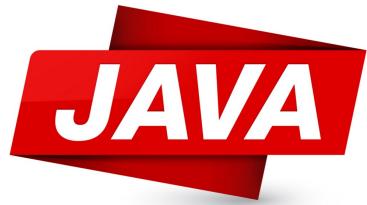
위 사진처럼 A댓글 이후 B댓글을 먼저 작성하였지만, 이후 A댓글에 A1댓글을 달면 B보다 위에 위치해야 한다. 따라서 댓글과 해당 댓글에 달린 대댓글은 하나로 묶어야 된다. 이를 위해 bundleId라는 컬럼을 추가해서 관리했다. 서비

வ <https://velog.io/@znftm97/%EC%8A%A4%ED%94%84%EB%A7%81%EC%97%90%EC%84%9C-%EB%8F%99%EC%8B%9C%EC%84%8B1-%EB%AC%B8%EC%A0%9C-%ED%95%B4%EA%B2%B0%ED%95%98%EA%B8%80-tl5imu04>

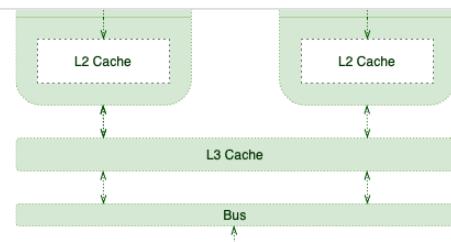
[Effective Java] 11장 동시성 + collection 유ти 메서드

아이템 78. 공유 중인 가변 데이터는 읽기 쓰기 모두 동기화해 사용하라 lock을 걸어서 sync하는건 성능에 좋지 않으니까, 원자적 데이터(boolean, long,...)를 읽고 쓸 때는 동기화하지 말아야겠다고 생각하기..

☞ <https://umbum.dev/1037>



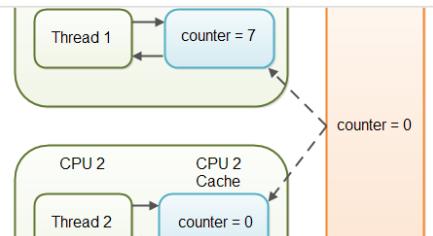
velog



자바 ConcurrentHashMap

즉 해쉬맵을 써드에 세이프하도록 만든 클래스. 하지만 HashMap과는 다르게 key, value에 null을 허용하지 않는다. ConcurrentHashMap은 concurrent multi-thread 환경에서 안정적으로 동작하고, HashTable과 synchronized

☜ <https://applefarm.tistory.com/100>



스레드의 동기화

멀티스레드 환경에서 A Thread가 작업하던 도중 B Thread에게 제어권이 넘겨갔을 때, A Thread가 사용하던 공유 자원을 B Thread가 임의로 변경했다면? A Thread가 다시 제어권을 받고 작업을 수행하면 의도와는 다른 결과를

வ <https://velog.io/@znftm97/%EC%8A%A4%ED%A0%88%EB%93%9C-%EB%8F%99%EA%B8%80%ED%99%94%EC%97%90-%EB%80%ED%95%B4%EC%84%9C-JAVA>

velog

논블로킹 알고리즘, Atomic 클래스와 CAS 연산

volatile, 가시성 문제, 동기화에 대한 이야기가 나오는데 해당 글을 참고하자.
동기화를 다루면서 Lock에 대해서 알아보았는데 문제점이 존재했다. Lock을
획득한 스레드 이외에 다른 모든 스레드들은 접근이 불가능하다는 것이다.

v <https://velog.io/@znftm97/%EB%85%BC%EB%B8%94%EB%A1%9C%ED%82%B9-%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-Atomic-%ED%81%B4%EB%9E%98%EC%8A%A4%EC%99%80-CAS-%EC%97%B0%EC%82%B0>

```
private static final long valueOffset;
68
69     |
70     | Records whether the underlying VM supports lockless compareAndSwap for longs. Whether
71     | compareAndSwapLong method works in either case, some constructions should be handled
72     | level to avoid locking user-visible locks.
73     |
74     static final boolean VM_SUPPORTS_LONG_CAS = VMSupportsCS();
75
76     static {
77         try {
78             valueOffset = unsafe.objectFieldOffset
79         }
```