Annotations in Spring Boot Redis

Description

Introduction

In this post I will explain the four most important annotation used when using Redis along with spring boot. By the end of this post you will get to know the annotations and its usage.

@EnableCaching

@EnableCaching annotation is used above the main class of the spring boot application which tells the Spring container that the caching feature will be used.

```
package com.springmicroservices;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cache.annotation.EnableCaching;

@SpringBootApplication
@EnableCaching
public class PetprojectRedisDemoApplication {
 public static void main(String[] args) {
  SpringApplication.run(PetprojectRedisDemoApplication.class, args);
 }
}
```

@Cacheable

@Cacheable annotation is used whenever we are performing GET operation or when we are retrieving the data from the database. We use this method above the method that is used to retrieve data from the DB. There are some attributes which we can use along with @Cacheable annotation.

In the below example, the returned value is cached in **UnicornEntity** and **id** is the unique key that identifies each entry in the cache.

```
    @Cacheable(value="UnicornEntity", key="#id")
    public UnicornEntity getUnicorn(long id) {
        Optional<UnicornEntity> unicorn = unicornRepository.findById(id);
        if(unicorn.isPresent()) {
            return unicorn.get();
        }
        return null;
    }
```

We can also perform caching based on a specific condition like below. If the length of firstname is less than 10 then the response will be cached.

```
@Cacheable(value="UnicornEntity", condition="#firstname.length<10")
```

@CachePut

@CachePut is a method level annotation which is used on update operation when we want to update the cache without interfering the method execution. The method will always execute and the results will be placed in the cache.

There is a small difference between @Cacheable and @CachePut annotation. @Cacheable skips the method execution and returns whatever is present in the cache, whereas @CachePut runs the method and puts the results in the cache.

```
@CachePut(cacheNames="UnicornEntity", key="#id")
public UnicornEntity updateUnicorn(long id, Unicorn unicorn) {
//code logic
}
```

@CacheEvict

@CacheEvict is a method level annotation which is used on delete operation when we want to remove stale or unused data from the cache. There are two forms of @CacheEvict.

Below form evicts all entries rather than one entry based on the key. This will clear all the entries in the cache *UnicornEntity*

```
@CacheEvict(value="UnicornEntity", allEntries=true)

Evict an entry by key:
```

```
@CacheEvict(key="#employee.emp_name")
```

Wrapping up

I hope you enjoyed the post and got to learn something out of it. Let me know your views. If you have any suggestions or doubts feel free to comment and I will be happy to answer. Keep learning ðŸ™‚

Category

1. Hands on
2. Spring Boot

Date Created
December 29, 2022
Author
kk-ravi144gmail-com