J2Cache实战

J2Cache介绍

两级的缓存框架。

L1：进程内的缓存：Caffeine/ehcache，类似于Map。

L2：集中式缓存：Redis/Memcached，

如果只用L1，程序重启，缓存会丢失。

如果只用L2，读写缓存，会造成网络的浪费。

J2Cache好处：通过对L1的读取，来降低对L2的读写次数。通过利用L2的持久化，防止程序重启缓存的丢失。

缓存的读取顺序：L1 > L2 > DB

操作

1。新建Spring boot项目。

2。引入J2Cache的jar包。

```
<dependency>
    <groupId>net.oschina.j2cache</groupId>
    <artifactId>j2cache-spring-boot2-starter</artifactId>
    <version>2.8.0-release</version>
</dependency>
<dependency>
    <groupId>net.oschina.j2cache</groupId>
    <artifactId>j2cache-core</artifactId>
    <version>2.8.0-release</version>
    <exclusions>
        <exclusion>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
        </exclusion>
```

```
            </exclusions>
        </dependency>
```

配置文件

application.yml

```
server:
  port: 8080

msb:
  redis:
    ip: 127.0.0.1
    port: 6379
    password:
    database: 0



spring:
  cache:
    type: redis
  redis:
    host: ${msb.redis.ip}
    password: ${msb.redis.password}
    port: ${msb.redis.port}

j2cache:
  open-spring-cache: true
  cache-clean-mode: passive
  allow-null-values: true
  #指定redis客户端使用lettuce，也可以使用Jedis
  redis-client: lettuce
  #开启二级缓存，false则表示只使用一级缓存
  l2-cache-open: true
  # 事件通知的机制，j2cache从1.3.0版本开始支持JGroups和Redis Pub/Sub两种方式进行缓存事件的通知。
  # 此处我们使用基于redis的发布订阅模式来通知缓存的各个节点来进行缓存数据的同步（由j2cache进行实现，我们写
  broadcast: net.oschina.j2cache.cache.support.redis.SpringRedisPubSubPolicy
  #  broadcast: jgroups
  #指定一级缓存提供者为caffeine
  L1:
    provider_class: caffeine
  #指定二级缓存提供者为redis
  L2:
    provider_class: net.oschina.j2cache.cache.support.redis.SpringRedisProvider
    config_section: lettuce
  sync_ttl_to_redis: true
```

```
  default_cache_null_object: false
  serialization: fst
caffeine:
  properties: /caffeine.properties   # 这个配置文件需要放在项目中
# lettuce是redis的一个客户端，也可以使用jedis，都是用来操作redis的java客户端
lettuce:
  mode: single
  namespace:
  storage: generic
  channel: j2cache
  scheme: redis
  hosts: ${msb.redis.ip}:${msb.redis.port}
  password: ${msb.redis.password}
  database: ${msb.redis.database}
  sentinelMasterId:
  maxTotal: 100
  maxIdle: 10
  minIdle: 10
  timeout: 10000
```

caffeine.properties

```
# 50是缓存的数量，2h是缓存的有效期。
testRegion=50,2h
```

核心操作类

```
package com.mashibing.j2cache;

import net.oschina.j2cache.CacheChannel;
import net.oschina.j2cache.CacheObject;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("/j2cache")
public class J2CacheController {

    private String key = "myKey";
    private String testRegion = "testRegion";
```

```
@Autowired
private CacheChannel cacheChannel;

@GetMapping("/getValue")
public String getValue(){
    CacheObject cacheObject = cacheChannel.get(testRegion,key);
    if (cacheObject.getValue() == null){
        // 从缓存取不到值，我们从数据库去查询
        String dbString = "数据库的值";
        // 放入缓存
        cacheChannel.set(testRegion,key, dbString);
        return dbString;

    }
    return cacheObject.getValue().toString();

}
@GetMapping("/getValue1")
public String getValue1(){
    CacheObject cacheObject = cacheChannel.get(testRegion,key);

    return cacheObject.getValue().toString();

}

/**
 * 一级缓存和二级缓存都会清理。
 * @return
 */
@GetMapping("/evict")
public String evict() {

    cacheChannel.evict(testRegion,key);
    return "evice success";
}

/**
 * 清理指定区域。
 * @return
 */
@GetMapping("/clear")
public String clear() {

    cacheChannel.clear(testRegion);
    return "clear success";
}
```

```
    /**
     * 判断缓存中的数据是否存在
     * @return
     */
    @GetMapping("/exists")
    public String exists(){
        boolean result = cacheChannel.exists(testRegion, key);
        return "exists:"+result;
    }

    /**
     * 判断数据从哪一级获取
     * @return
     */
    @GetMapping("/check")
    public String check(){
        int level  = cacheChannel.check(testRegion, key);
        return "level:"+level;
    }

}
```

实际中：根据自己实际的key和value，以及业务，做对应的设置和删除操作。