一、Redis介绍

Redis是当前比较热门的NOSQL系统之一，它是一个开源的使用ANSI c语言编写的key-value存储系统（区别于MySQL的二维表格的形式存储。）。和Memcache类似，但很大程度补偿了Memcache的不足。和Memcache一样，Redis数据都是缓存在计算机内存中，不同的是，Memcache只能将数据缓存到内存中，无法自动定期写入硬盘，这就表示，一断电或重启，内存清空，数据丢失。所以Memcache的应用场景适用于缓存无需持久化的数据。而Redis不同的是它会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，实现数据的持久化。

Redis的特点：

1，Redis读取的速度是110000次/s，写的速度是81000次/s；

2，原子 。Redis的所有操作都是原子性的，同时Redis还支持对几个操作全并后的原子性执行。

3，支持多种数据结构：string（字符串）；list（列表）；hash（哈希），set（集合）；zset(有序集合)

4，持久化，集群部署

5，支持过期时间，支持事务，消息订阅

Spring Cache 是一个非常优秀的缓存组件。自Spring 3.1起，提供了类似于@Transactional注解事务的注解Cache支持，且提供了Cache抽象，方便切换各种底层Cache（如：redis）

使用Spring Cache的好处：

1，提供基本的Cache抽象，方便切换各种底层Cache；

2，通过注解Cache可以实现类似于事务一样，缓存逻辑透明的应用到我们的业务代码上，且只需要更少的代码就可以完成；

3，提供事务回滚时也自动回滚缓存；

4，支持比较复杂的缓存逻辑；

二、数据字典模块添加Redis缓存

1、common_utils模块，添加redis依赖

```xml
<!-- redis -->
<dependency>
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>

<!-- spring2.X集成redis所需common-pool2-->
<dependency>
     <groupId>org.apache.commons</groupId>
     <artifactId>commons-pool2</artifactId>
     <version>2.6.0</version>
</dependency>
```

2、common_utils模块，添加Redis配置类

```java
import com.fasterxml.jackson.annotation.JsonAutoDetect;
import com.fasterxml.jackson.annotation.PropertyAccessor;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.springframework.cache.CacheManager;
import org.springframework.cache.annotation.EnableCaching;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.cache.RedisCacheConfiguration;
import org.springframework.data.redis.cache.RedisCacheManager;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.serializer.Jackson2JsonRedisSerializer;
import org.springframework.data.redis.serializer.RedisSerializationContext;
import org.springframework.data.redis.serializer.RedisSerializer;
import org.springframework.data.redis.serializer.StringRedisSerializer;

import java.time.Duration;

@Configuration
@EnableCaching
public class RedisConfig {

    /**
     * 设置RedisTemplate规则
```

```
     * @param redisConnectionFactory
     * @return
     */
    @Bean
    public RedisTemplate<Object, Object> redisTemplate(RedisConnectionFactory
        RedisTemplate<Object, Object> redisTemplate = new RedisTemplate<>();
        redisTemplate.setConnectionFactory(redisConnectionFactory);
        Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new Jackson

//解决查询缓存转换异常的问题
        ObjectMapper om = new ObjectMapper();
// 指定要序列化的域，field,get和set,以及修饰符范围，ANY是都有包括private和public
        om.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY)
// 指定序列化输入的类型，类必须是非final修饰的，final修饰的类，比如String,Integer
        om.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);
        jackson2JsonRedisSerializer.setObjectMapper(om);

//序列号key value
        redisTemplate.setKeySerializer(new StringRedisSerializer());
        redisTemplate.setValueSerializer(jackson2JsonRedisSerializer);
        redisTemplate.setHashKeySerializer(new StringRedisSerializer());
        redisTemplate.setHashValueSerializer(jackson2JsonRedisSerializer);

        redisTemplate.afterPropertiesSet();
        return redisTemplate;
    }

    /**
     * 设置CacheManager缓存规则
     * @param factory
     * @return
     */
    @Bean
    public CacheManager cacheManager(RedisConnectionFactory factory) {
        RedisSerializer<String> redisSerializer = new StringRedisSerializer()
        Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new Jackson

//解决查询缓存转换异常的问题
        ObjectMapper om = new ObjectMapper();
        om.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY)
        om.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);
        jackson2JsonRedisSerializer.setObjectMapper(om);

```

```
// 配置序列化（解决乱码的问题）,过期时间600秒
        RedisCacheConfiguration config = RedisCacheConfiguration.defaultCache
                .entryTtl(Duration.ofSeconds(600))
                .serializeKeysWith(RedisSerializationContext.SerializationPai
                .serializeValuesWith(RedisSerializationContext.SerializationP
                .disableCachingNullValues();

        RedisCacheManager cacheManager = RedisCacheManager.builder(factory)
                .cacheDefaults(config)
                .build();
        return cacheManager;
    }
}
```

3、在service_cmn模块，配置文件添加redis配置

```
spring.redis.host=192.168.44.165
spring.redis.port=6379
spring.redis.database= 0
spring.redis.timeout=1800000

spring.redis.lettuce.pool.max-active=20
spring.redis.lettuce.pool.max-wait=-1
#最大阻塞等待时间(负数表示没限制)
spring.redis.lettuce.pool.max-idle=5
spring.redis.lettuce.pool.min-idle=0

```

4、通过注解添加redis缓存

(1) 缓存@Cacheable

根据方法对其返回结果进行缓存，下次请求时，如果缓存存在，则直接读取缓存数据返回；如果缓存不存在，则执行方法，并把返回的结果存入缓存中。一般用在查询方法上。

查看源码，属性值如下：

属性/方法名	解释
value	缓存名，必填，它指定了你的缓存存放在哪块命名空间
cacheNames	与 value 差不多，二选一即可
key	可选属性，可以使用 SpEL 标签自定义缓存的key

(2) 缓存@CachePut

使用该注解标志的方法，每次都会执行，并将结果存入指定的缓存中。其他方法可以直接从响应的缓存中读取缓存数据，而不需要再去查询数据库。一般用在新增方法上。

查看源码，属性值如下：

属性/方法名	解释
value	缓存名，必填，它指定了你的缓存存放在哪块命名空间
cacheNames	与 value 差不多，二选一即可
key	可选属性，可以使用 SpEL 标签自定义缓存的key

(3) 缓存@CacheEvict

使用该注解标志的方法，会清空指定的缓存。一般用在更新或者删除方法上

查看源码，属性值如下：

属性/方法名	解释
value	缓存名，必填，它指定了你的缓存存放在哪块命名空间
cacheNames	与 value 差不多，二选一即可
key	可选属性，可以使用 SpEL 标签自定义缓存的key
allEntries	是否清空所有缓存，默认为 false。如果指定为 true，则方法调用后将立即清空所有的缓存

属性/方法名	解释
beforeInvocation	是否在方法执行前就清空，默认为 false。如果指定为 true，则在方法执行前就会清空缓存

5、查询数据字典列表添加Redis缓存

```
//根据数据id查询子数据列表
@Cacheable(value = "dict", key = "'selectIndexList'+#id")
@Override
public List<Dict> findChlidData(Long id) {
    QueryWrapper<Dict> wrapper = new QueryWrapper<>();
    wrapper.eq("parent_id",id);
    List<Dict> dictList = baseMapper.selectList(wrapper);
    //向list集合每个dict对象中设置hasChildren
    for (Dict dict:dictList) {
        Long dictId = dict.getId();
        boolean isChild = this.isChildren(dictId);
        dict.setHasChildren(isChild);
    }
    return dictList;
}
```