Redis学习笔记

Redis简介

Redis是一种数据库,是能够存储数据、管理数据的软件。 注意：不同的数据库适合存储的数据不同，流行的数据库都要学习，它们有不同的特点。甚至在一个项目中会使用多种数据库。 Redis解决了mysql,oracle不能解决的问题。

关系型数据库都是以表为单位存储的，如oracle,mysql,db2,sql server...。数据量大时表与表之间的关系复杂，存储量大。 非关系型数据库：彻底改变底层存储机制，不再擦词用关系数据模型，而是采用聚合数据结构存储数据，如redis,mongoDB,HBase...

Redis(Remote Directory Server，远程字典服务器），是一个用C语言编写的、开源的、基于内存运行并支持持久化的（redis中的数据大部分时间都是存储在内存中的，访问效率高，适合存储简单、少量、经常访问的数据）、高性能的NoSQL数据库，也是当前热门的NoSQL数据库之一。Redis又被当成"缓存数据库"，但会定期持久化。

Redis的特点:

1.支持数据持久化 Redis可以将内存中的数据保存在磁盘中，重启的时候可以再次加载进行使用。2.支持多种数据结构 Redis不仅仅支持简单的key-value类型的数据，同时还提供list,set,zset,hash等数据结构的存储。 3.支持数据备份 Redis支持数据的备份，即master-slave模式的数据备份。

Redis的环境：

- Windows版本：https://github.com/MSOpenTech/redis/releases
- Linux版本：https://download.redis.io/releases

Linux系统中安装Redis：

1. 将安装包上传到Linux系统并进行解压;
2. 安装Redis的依赖环境gcc，命令为`yum install gcc-c++`;
3. 进入安装包解压后的文件，进行编译，命令`make`;
4. 进入Redis的src目录，进行安装，命令为：`make install`;

Note： Windows上直接下载下来进行解压就行了，无需安装!

Redis服务启动与停止

- Linux中redis服务启动可以使用redis-server，默认端口为6379；
- `Ctrl+C` 停止redis服务；
- 修改配置文件redis.conf,是的redis进入后台运行，避免霸屏模式，将配置文件 `daemonize` 参数修改为 `yes` 即可，将参数 `requirepass` 前的 `#` 删掉代表着需要密码验证登录 `requirepass` 参数后面的字符串代表着默认密码，我们可以修改成自己的密码，注释掉 `bind 127.0.0.1` 参数即可实现远程客户端连接；在Windows电脑上远程连接Redis，在Redis文件夹下打开powershell命令行工具使用 `.\redis-cli.exe -h 地址 -p 端口 -a 密码` 进行连接；
- 使用命令 `src/redis-server ./redis.conf` 启动服务；
- 修改密码后重新启动服务，启动客服端， 使用命令 `src/redis-cli -a 具体的密码` 或者客户端启动会输入命令 `author 具体的密码` 进行登录；

Note： 在Windows系统中直接双击 `redis-server.exe` 和 `redis-cli.exe` 两个文件即可；

Redis的数据类型

介绍

Redis存储的是key-value结构的数据，其中key是字符串类型，value有五种常用的数据类型

- 字符创 string
- 哈希 hash
- 列表 list
- 集合 set
- 有序集合 sorted set

Redis常用命令

字符串string操作命令

- SET key vlaue 设置指定key 的值
- GET key 获取指定key的值
- SETEX key secends value 设置指定key的值并将key的过期时间设为seconds秒
- SETNX key value 只有key不存在的情况下设置key的值；

更多的命令参考Redis中文网的教程；

哈希hash操作命令

Redis hash是一个string类型的field和value的映射表，hash特别适合用于存储对象，常用命令：

- HSET key field value 将哈希表key中的字段filed的值设置为value
- HGET key field 获取存储在哈希表中指定字段
- HDEL key field 删除存储在哈希表中的指定字段
- HKEYS key 获取哈希表中所有字段
- HVALS key 获取哈希表中的所有值
- HGETALL key 获取在哈希表中的指定key的所有字段和值

列表list操作命令

Redis列表是简单的字符串列表，按照插入点顺序排序，常用命令：

- LPUSH key value1 [value2] 将一个或者多个值插入到列表头部
- LRANGE key start stop 获取列表指定范围内的元素
- RPOP key 移除并获取列表最后一个元素
- LLEN key 获取指定列表的长度
- BRPOP key1 [key2] timeout 移除并获取列表的最后一个元素，如果列表没有元素会阻塞列表直到等待超时或者发现可弹出元素为止

集合set操作命令

Redis set 是string类型的无序集合，集合成员是唯一的，意味着集合里面不能有重复的元素

- SADD key member1 [member2] 向集合中添加元素、
- SMEMBERS key 返回集合中的所有成员
- SCARD key 获取集合中的成员数
- SINTER key1 [key2] 返回给定所有集合的交集
- SUNION key1 [key2] 返回所有给定集合的并集

- SDIFF key1 [key2] 返回给定所有集合的差集
- SREM key member1 [member2] 移除集合中的一个或者多个成员

Redis常用命令

- KEYS pattern 查找所有符合给定pattern的key
- EXISTS key 检查给定key是否存在
- TYPE key 返回key所存储的值的类型
- TTL key 返回给定key的剩余生存时间，以秒为单位
- DEL key 改命令用于key存在时删除key

在Java中操作Redis

在普通的maven项目中操作Redis

在maven项目中引入依赖

```
<dependency>
            <groupId>redis.clients</groupId>
            <artifactId>jedis</artifactId>
            <version>2.8.0</version>
</dependency>
```

创建一个测试类

```java
public class JedisTest {
    @Test
    public void testRedis(){
        // 获取连接
        Jedis jedis = new Jedis("localhost",6379);

        // 执行具体的操作
        jedis.set("username","xiaoming");

        String username = jedis.get("username");
        System.out.println(username);

        jedis.del("username");  //删除key

        jedis.hset("myhash","addr","chongqing");
        System.out.println(jedis.hget("myhash", "addr"));

        System.out.println(jedis.keys("*"));
        // 关闭连接
        jedis.close();
    }
}
```

在springboot项目中操作Redis

导入相关坐标

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

spring Data Redis中提供了一个高度封装的类：RedisTemplate，针对jedis客户端中大量的api进行了归类封装，将同一类操作封装为operation接口

配置yml文件

```yaml
spring:
  redis:
    host: localhost
    port: 6379
    database: 0
    jedis:
      # 配置Redis连接池
      pool:
        max-active: 8  # 最大连接数
        max-wait: 1ms # 连接池最大阻塞等待时间
        max-idle: 8 # 连接池最大空闲连接
        min-idle: 0 # 连接池最小空闲连接
```

新建一个配置类 编写序列化器

```java
@Configuration
public class redisConfig extends CachingConfigurerSupport {
    @Bean
    public RedisTemplate<Object,Object> redisTemplate(RedisConnectionFactory
connectionFactory){

        RedisTemplate<Object,Object> redisTemplate =new RedisTemplate<>();

        redisTemplate.setKeySerializer(new StringRedisSerializer());
        redisTemplate.setHashKeySerializer(new StringRedisSerializer());

        redisTemplate.setConnectionFactory(connectionFactory);

        return redisTemplate;
    }

}
```

编写一个测试类测试相关方法

```java
package com.exampl.demo;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.data.redis.connection.DataType;
import org.springframework.data.redis.core.*;
import org.springframework.test.context.junit4.SpringRunner;

import java.util.List;
import java.util.Set;
import java.util.concurrent.TimeUnit;

@SpringBootTest
@RunWith(SpringRunner.class)
public  class DemoApplicationTests {

    @Autowired
    private RedisTemplate redisTemplate;

    // 1.进行string类型数据操作

    @Test
    public void testString(){

        /**
         * @description: 操作string 类型的数据
         * @author: song
         * @date: 2022/7/18 19:40
         * @param: []
         * @return: void
         **/

        redisTemplate.opsForValue().set("cityone","chongqing");
        String  cityone = (String) redisTemplate.opsForValue().get("cityone");
        System.out.println(cityone);

        redisTemplate.opsForValue().set("key1", "value1", 10l, TimeUnit.SECONDS);

        Boolean aBoolean = redisTemplate.opsForValue().setIfAbsent("cityone",
"beijing");
        System.out.println(aBoolean);

    }
    /**
     * @description:操作hash类型的数据
     * @author: song
     * @date: 2022/7/18 19:41
     * @param:
```

```
     * @return:
     **/

    @Test
    public void testHash(){
        HashOperations hashOperations = redisTemplate.opsForHash();
        hashOperations.put("002","name","xiaoming");
        hashOperations.put("002","age","20");
        hashOperations.put("002" ,"address","beijing");

        System.out.println(hashOperations.get("002", "age"));

        // 获取hash结构中的所有字段
        System.out.println(hashOperations.keys("002"));
        List values = hashOperations.values("002");
        for (Object value : values) {
            System.out.println(values);
        }
    }

    @Test
    public void testList(){

        ListOperations listOperations = redisTemplate.opsForList();

        listOperations.leftPush("mylist", "a");
        listOperations.leftPushAll("mylist","a","c","b");

        List<String> mylist = listOperations.range("mylist", 0, -1);
        for (String s : mylist) {
            System.out.println(s);
        }

        Long size = listOperations.size("mylist");
        int intValue = size.intValue();
        for (int i = 0; i < intValue; i++) {
            String mylist1 = (String) listOperations.rightPop("mylist");
            System.out.println(mylist1);
        }
    }

    @Test
    public void testSet(){
        SetOperations setOperations = redisTemplate.opsForSet();
        setOperations.add("myset","a","b","c","a");

        Set myset = setOperations.members("myset");
        for (Object o : myset) {
            System.out.println(o);
        }
```

```
        setOperations.remove("myset","a","b");
    }

    @Test
    public void testZset(){
        ZSetOperations zSetOperations = redisTemplate.opsForZSet();

        zSetOperations.add("zset","a",10.0);
        zSetOperations.add("zset","b",11.0);
        zSetOperations.add("zset","c",12.0);
        zSetOperations.add("zset","a",13.0);

        Set myset = zSetOperations.range("myset", 0, -1);
        for (Object o : myset) {
            System.out.println(o);
        }
        zSetOperations.incrementScore("zset","b", 20.0);

        zSetOperations.remove("zset","a","c");
    }

    @Test
    public void testCommon(){
        // 获取redis中所有的key
        Set keys = redisTemplate.keys("*");
        for (Object key : keys) {
            System.out.println(key);
        }
        // 判断某个key是否存在
        Boolean nid_mingz = redisTemplate.hasKey("nid mingz");
        System.out.println(nid_mingz);
        // 删除指定的key
        redisTemplate.delete(nid_mingz);
        // 获取指定key对应的value类型
        DataType zset = redisTemplate.type("zset");
        System.out.println(zset.name());
    }
}
```