作者:Wilson

Java连接Redis集群

一、导入POM.XML

SpringBoot2.0 Redis相关Jar包

```
  <!--默认是lettuce客户端-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-redis</artifactId>
        </dependency>

        <!-- redis依赖commons-pool 这个依赖一定要添加 -->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-pool2</artifactId>
        </dependency>
```

二、编写application.yml

```
spring:
  redis:
    password: guoweixin  #密码
    lettuce:  #lettuce连接池配置
      pool:
        max-active: 8
        max-idle: 8
        min-idle: 0
        max-wait: 1000
      shutdown-timeout: 100
    cluster:  #集群配置
      nodes:
        - 192.168.46.136:7000
        - 192.168.46.136:7001
        - 192.168.46.136:7002
        - 192.168.46.136:7003
        - 192.168.46.136:7004
        - 192.168.46.136:7005
      max-redirects: 3
```

三、RedisConfig编写

前面单机访问整合一样(SpringBoot2+Data-Redis)

```
/**
 * @ClassName RedisConfig
 * @Description TODO
 * @Author guoweixin
 * @Version 1.0
 */
@Configuration
@Log
public class RedisConfig extends CachingConfigurerSupport {

    /**
     * 自定义缓存key的生成策略。默认的生成策略是看不懂的(乱码内容) 通过Spring 的依赖注入特性进行自定义的配置注入并且此类是一个配置类可以更多程度的自定义配置
     *
     * @return
     */
    @Bean
    @Override
    public KeyGenerator keyGenerator() {
        return new KeyGenerator() {
            @Override
            public Object generate(Object target, Method method, Object... params) {
                StringBuilder sb = new StringBuilder();
                sb.append(target.getClass().getName());
                sb.append(method.getName());
                for (Object obj : params) {
                    sb.append(obj.toString());
                }
                log.info("打印："+sb.toString());
                return sb.toString();
            }
        };
    }

    /**
     * 缓存配置管理器
     */
    @Bean
    public CacheManager cacheManager(LettuceConnectionFactory factory) {
        //以锁写入的方式创建RedisCacheWriter对象
        RedisCacheWriter writer = RedisCacheWriter.lockingRedisCacheWriter(factory);
        //创建默认缓存配置对象
        RedisCacheConfiguration config = RedisCacheConfiguration.defaultCacheConfig();
        RedisCacheManager cacheManager = new RedisCacheManager(writer, config);
        return cacheManager;
    }



    @Bean
    public RedisTemplate<String,Object> redisTemplate(LettuceConnectionFactory
factory){
        RedisTemplate<String,Object> template = new RedisTemplate <>();
        template.setConnectionFactory(factory);


        Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new Jackson2JsonRedisSerializer(Object.class);
        ObjectMapper om = new ObjectMapper();
        om.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
        om.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);
        jackson2JsonRedisSerializer.setObjectMapper(om);

        StringRedisSerializer stringRedisSerializer = new StringRedisSerializer();

        // 在使用注解@Bean返回RedisTemplate的时候，同时配置hashKey与hashValue的序列化方式。
        // key采用String的序列化方式
        template.setKeySerializer(stringRedisSerializer);
        // value序列化方式采用jackson
        template.setValueSerializer(jackson2JsonRedisSerializer);

        // hash的key也采用String的序列化方式
        template.setHashKeySerializer(stringRedisSerializer);
        // hash的value序列化方式采用jackson
        template.setHashValueSerializer(jackson2JsonRedisSerializer);
        template.afterPropertiesSet();
        return template;
    }

}
```

四、Redis Cluster集群创建

集群创建时，如果是让JAVA远程连接Redis Cluster，则需要写明IP地址（不可以用127.0.0.1)

```
#原始Redis:
/usr/local/redis_cluster/src/redis-cli --cluster  create  -a guoweixin 192.168.20.138:6381 192.168.20.138:6382 192.168.20.138:6383 192.168.20.138:6384 192.168.20.138:6385 192.168.20.138:6386 --cluster-replicas 1
```

```
#Docker方式:
docker exec -it redis-6381 redis-cli --cluster  create  -a guoweixin 192.168.20.138:6381 192.168.20.138:6382 192.168.20.138:6383 192.168.20.138:6384 192.168.20.138:6385 192.168.20.138:6386 --cluster-replicas 1
```

如果无法创建新集群，需要将上次集群生成的/data文件进行删除。重新创建集群即可

作者:Wilson

五、开放防火墙端口

6381-6386依次开放端口（如下命令只针对Centos7以上）

查看已经开放的端口：

```
firewall-cmd --list-ports
```

开启端口:

```
    firewall-cmd --zone=public --add-port=6381/tcp --permanent
```

重启防火墙:

```
firewall-cmd --reload #重启
firewall systemctl stop firewalld.service #停止
firewall systemctl disable firewalld.service #禁止firewall开机启动
```