redis整合springboot

2020年2月15日 11:21

1.用户登录的三个遗留问题

1.1业务功能问题

1.2new Jedis

代码性能低。有了连接池，如何整合到springboot?

1.3redis单节点

可以使用目前已有的3个redis集群实现分布式

2.整合步骤

2.1回顾springboot自动配置原理

@Configuration：类注解，表示一个类代表配置逻辑

@Bean：方法注解，在一个配置类中，实现对象框架维护

2.2spring框架维护连接池对象

目的：在业务层注入连接池对象，拿到资源，使用完毕之后还回资源。整个框架系统只会创建一次连接池。

2.3配置步骤

a. 编写一个配置类,添加配置注解

```
@Configuration
public class ShardedJedisConfig {
 }
```

相当于框架系统一旦加载这个类,就加载了一个xml配置文件

b. 在配置类中,实现一个连接池对象的创建

在方法中new一个连接池对象,注入使用

```
@Bean
    public ShardedJedisPool initPool(){
        //收集节点信息
        List<JedisShardInfo> info=new ArrayList<>();
        info.add(new JedisShardInfo("10.9.104.184",6379));
        info.add(new JedisShardInfo("10.9.104.184",6380));
        info.add(new JedisShardInfo("10.9.104.184",6381));
        //配置属性config
        JedisPoolConfig config=new JedisPoolConfig();
        config.setMaxTotal(200);
        config.setMinIdle(3);
        config.setMaxIdle(8);
        return new ShardedJedisPool(config,info);
    }
```

3.利用连接池对象修改用户系统代码

之前是new Jedis.现在有了连接池对象(分片连接池),可以注入连接池,调用业务逻辑中获取资源,用完了还回资源.

3.1登录逻辑修改

```
@Autowired
    private ShardedJedisPool pool;
    public String doLogin(User user){
        //获取一个连接池资源
        ShardedJedis jedis = pool.getResource();
        String ticket="";
        user.setUserPassword(MD5Util.md5(user.getUserPassword()));
        User exist=um.selectUserByNameAndPw(user);//select * from t_user where
name= and pw=
        if(exist==null){
            return ticket;
        }else{
         exist.setUserPassword(null);
            ObjectMapper om= MapperUtil.MP;
            //Jedis jedis=new Jedis("10.9.104.184",6380);
            try{
                String uJson = om.writeValueAsString(exist);
              ticket="EM_TICKET"+System.currentTimeMillis()+user.getUserName();
                jedis.setex(ticket,60*60*2,uJson);
            }catch(Exception e){
                e.printStackTrace();
                return "";
            }finally {
                /*if(jedis!=null){
                    jedis.close();
                }*/
                if(jedis!=null){
                    pool.returnResource(jedis);
                }
            }
            return ticket;
        }
    }
```

3.2用户状态获取

```
        //根据ticket查询redis中数据
    public String queryUserData(String ticket){
        //从连接池获取资源
        ShardedJedis jedis = pool.getResource();
        //Jedis jedis=new Jedis("10.9.104.184",6380);
        try{
            return jedis.get(ticket);
        }catch(Exception e){
            e.printStackTrace();
            return null;
        }finally {
            if(jedis!=null){
                //jedis.close();
```

```
            pool.returnResource(jedis);
        }
    }
}
```

3.4验证是否连接池生效

保证redis集群启动着的，否则创建的连接资源无法访问redis

3.5读取属性的配置逻辑

@ConfigurationProperties：配合配置类实现在配置类中的属性赋值的功能。

@Value：也可以给属性赋值，但是功能么有ConfigrationProperties强大。

a. 在配置类上，添加@ConfiugrationProperties

```
@Configuration
@ConfigurationProperties(prefix = "redis")//当前配置类主要配置redis
public class ShardedJedisConfig {
    private Integer maxTotal;//properties文件中 redis.maxTotal
    private Integer maxIdle;//properties文件中 redis.maxIdle;
    private Integer minIdle;
    //ConfigurationProperties支持 属性值以，隔开赋值list类型数据
    private List<String> nodes;//有多少个节点就以10.9.9.9:6379，10.9.9.9：6380,
    //在properties文件中，需要按照对应关系去指定配置key值
    //key=prefix.属性名称
GETTER&&SETTER
```

b. application.properties准备这些数据

```
#redis的分片连接池属性值
easymall.redis.maxTotal=200
easymall.redis.maxIdle=8
easymall.redis.minIdle=3
easymall.redis.nodes=10.9.104.184:6379,10.9.104.184:6380,10.9.104.184:6381
```

c. @Bean注解的方法内容代码修改

使用属性值，创建对应对象

```
 @Bean
    public ShardedJedisPool initPool(){
        //收集节点信息
        List<JedisShardInfo> info=new ArrayList<>();
        //从nodes数据中解析出来每一个节点ip地址和端口号
        //nodes={"10.9.104.184:6379","10.9.104.184:6380","10.9.104.184:6381"}
        for (String node:nodes) {
            //对list的所有元素进行循环，每次循环拿出一个元素值，赋值给node
            //node循环第一次 node="10.9.104.184:6379"
            String host=node.split(":")[0];
            int port=Integer.parseInt(node.split(":")[1]);//"6379"-->6379
            info.add(new JedisShardInfo(host,port));
        }
        //配置属性config
        JedisPoolConfig config=new JedisPoolConfig();
        config.setMaxTotal(maxTotal);
```

```
        config.setMinIdle(minIdle);
        config.setMaxIdle(maxIdle);
        return new ShardedJedisPool(config,info);
    }
```

总结整合redis技术到springboot

目的：所有通过代码连接技术实现操作技术发送命令的过程，都需要在springboot以配置类的形式，创建想要的各种对象（jedis 创建ShardedJedisPool）。然后在业务逻辑中注入使用这些对象。

配置类基本结构

@Configuration标识一个自定义的配置类，在类中实现配置逻辑

@Bean 配置类中创建框架管理bean对象注解，但是bean对象很多在初始化时，都需要参数赋值。

@ConfigurationProperties：

在配置类中的私有属性配合getter&&setter方法实现带有前缀匹配的赋值，例如properties中 easymall.redis.nodes的key值，在代码中可以定义前缀为easymall.redis，属性名称nodes