**[springboot操作redis](https://www.cnblogs.com/nfcm/p/7833032.html)**

springboot整合redis有两种方式，分别是jedis和RedisTemplate，而RedisTemplate是SpringDataRedis中对JedisApi的高度封装

### **第一步，引入springboot redis的maven依赖**

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-redis</artifactId>

</dependency>

**创建application.properties文件**

#Redis服务器地址

spring.redis.host=152.136.37.224

#Redis服务器连接端口

spring.redis.port=6379

#Redis数据库索引（默认为0）

spring.redis.database=0

#连接池最大连接数（使用负值表示没有限制）

spring.redis.jedis.pool.max-active=50

#连接池最大阻塞等待时间（使用负值表示没有限制）

spring.redis.jedis.pool.max-wait=3000

spring.redis.password=068`

#连接池中的最大空闲连接

spring.redis.jedis.pool.max-idle=20

#连接池中的最小空闲连接

spring.redis.jedis.pool.min-idle=2

#连接超时时间（毫秒）

spring.redis.timeout=5000

**RedisTemplate模板介绍**

spring 封装了 RedisTemplate 对象来进行对redis的各种操作，它支持所有的 redis 原生的 api。

RedisTemplate中定义了5种数据结构的操作。

redisTemplate.opsForValue();//操作字符串

redisTemplate.opsForHash();//操作hash

redisTemplate.opsForList();//操作list

redisTemplate.opsForSet();//操作set

redisTemplate.opsForZSet();//操作有序set

**key和value序列化**

spring-data-redis的序列化类有下面这几个:

GenericToStringSerializer: 可以将任何对象泛化为字符串并序列化

Jackson2JsonRedisSerializer: 跟JacksonJsonRedisSerializer实际上是一样的

JacksonJsonRedisSerializer: 序列化object对象为json字符串

JdkSerializationRedisSerializer: 序列化java对象

StringRedisSerializer: 简单的字符串序列化

**template(模版)**

Spring Data Redis提供了两个模板：

RedisTemplate：默认使用JdkSerializationRedisSerializer: 序列化java对象，可以保存对象等

StringRedisTemplate：默认使用StringRedisSerializer: 字符串序列化，可以用来存储string格式

首先我们先创建一个RedisTemplate模板类，类型的key是String类型，value是Object类型（如果key和value都是String类型，建议使用StringRedisTemplate）

@Bean

public RedisTemplate redisTemplate(RedisConnectionFactory factory){

//创建一个模板类

RedisTemplate<String, Object> template = new RedisTemplate<String, Object>();

//将刚才的redis连接工厂设置到模板类中

template.setConnectionFactory(factory);

return template;

}

然后在创建一个StringTemplate模板类

@Bean

public StringRedisTemplate stringRedisTemplate(RedisConnectionFactory factory){

//创建一个StringRedisTemplate模板类

StringRedisTemplate stringRedisTemplate=new StringRedisTemplate();

//将刚才的redis连接工厂设置到模板类中

stringRedisTemplate.setConnectionFactory(factory);

return stringRedisTemplate;

}

进行单元测试

@RunWith(SpringJUnit4ClassRunner.class)

@SpringBootTest(classes = Application.class)public class RedisTest {

@Autowired

RedisConnectionFactory factory;

@Test

public void testRedis(){

//得到一个连接

RedisConnection conn = factory.getConnection();

conn.set("hello".getBytes(), "world".getBytes());

System.out.println(new String(conn.get("hello".getBytes())));

}

}

当保存一条数据的时候，key和value都要被序列化成json数据，取出来的时候被序列化成对象，key和value都会使用序列化器进行序列化，spring data redis提供多个序列化器

* GenericToStringSerializer：使用Spring转换服务进行序列化；
* JacksonJsonRedisSerializer：使用Jackson 1，将对象序列化为JSON；
* Jackson2JsonRedisSerializer：使用Jackson 2，将对象序列化为JSON；
* JdkSerializationRedisSerializer：使用Java序列化；
* OxmSerializer：使用Spring O/X映射的编排器和解排器（marshaler和unmarshaler）实
* 现序列化，用于XML序列化；
* StringRedisSerializer：序列化String类型的key和value。

RedisTemplate会默认使用JdkSerializationRedisSerializer，这意味着key和value都会通过Java进行序列化。StringRedisTemplate默认会使用StringRedisSerializer

例如，假设当使用RedisTemplate的时候，我们希望将Product类型的value序列化为JSON，而key是String类型。RedisTemplate的setKeySerializer()和setValueSerializer()方法就需要如下所示：

@Bean

public RedisTemplate redisTemplate(RedisConnectionFactory factory) {

// 创建一个模板类

RedisTemplate<String, Object> template = new RedisTemplate<String, Object>();

// 将刚才的redis连接工厂设置到模板类中

template.setConnectionFactory(factory);

// 设置key的序列化器

template.setKeySerializer(new StringRedisSerializer());

// 设置value的序列化器

template.setKeySerializer(new StringRedisSerializer())

//使用Jackson 2，将对象序列化为JSON

Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new Jackson2JsonRedisSerializer(Object.class);

//json转对象类，不设置默认的会将json转成hashmap

ObjectMapper om = new ObjectMapper();

om.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);

om.enableDefaultTyping(ObjectMapper.DefaultTyping.NON\_FINAL);

jackson2JsonRedisSerializer.setObjectMapper(om);

template.setValueSerializer(jackson2JsonRedisSerializer);

return template;

}