1. java使用redis

很简单，看我的github的例子就行

1. spirng整合redis，这里用xml配置方式实现

2.1导包

<dependency>  
 <groupId>org.springframework.data</groupId>  
 <artifactId>spring-data-redis</artifactId>  
 <version>1.0.2.RELEASE</version>  
</dependency>  
<dependency>  
 <groupId>redis.clients</groupId>  
 <artifactId>jedis</artifactId>  
 <version>2.1.0</version>  
</dependency>

2.2配置applicationContext.xml

Spring提供了很多序列化器

Jackson2JsonRedisSerializer

JdkSerializationRedisSerializer

OxmSerializer

StringRedisSerializer

GenericToStringRedisSerializer

GenericJackson2JsonRedisSerializer

*<!--redisconfig-->*<bean id="poolConfig" class="redis.clients.jedis.JedisPoolConfig">  
 <property name="maxIdle" value="50" />  
 <property name="maxActive" value="100" />  
 <property name="maxWait" value="6000"/>  
</bean>  
*<!--rdis连接工厂-->*<bean id="connectionFactory" class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory">  
 <property name="hostName" value="localhost" />  
 <property name="port" value="6379" />  
 <property name="poolConfig" ref="poolConfig" />  
</bean>  
  
*<!--string 和jdk序列化器-->*<bean id="jdkSerializationRedisSerializer" class="org.springframework.data.redis.serializer.JdkSerializationRedisSerializer" />  
<bean id="stringRedisSerializer" class="org.springframework.data.redis.serializer.StringRedisSerializer" />  
  
*<!--配置redisTemplate-->*<bean id="redisTemplate" class="org.springframework.data.redis.core.RedisTemplate">  
 <property name="connectionFactory" ref="connectionFactory" />  
 *<!--key和value的序列化方式 常用string或者jdk-->* <property name="keySerializer" ref="stringRedisSerializer" />  
 <property name="valueSerializer" ref="stringRedisSerializer" />  
</bean>

2.3测试

ApplicationContext applicationContext = new ClassPathXmlApplicationContext(  
 "applicationContext.xml");  
RedisTemplate redisTemplate = applicationContext  
 .getBean(RedisTemplate.class);

* 1. StringRedisTemplete和RedisTemplete的区别

StringRedisTemplete继承RedisTemplete，内部使用string序列化器，所以适合操作key和value都是string的数据

3 Spring中发布订阅

3.1首先提供接收消息的类，它将实现 org.springframework.data.redis.connection.MessageListener 接口，并实现接口定义的方法 public void onMessage（Message message,byte[]pattern）

/\*\*\* imports \*\*\*/

1. **public** **class** RedisMessageListener **implements** MessageListener {
2. **private** RedisTemplate redisTemplate;
3. /\*\*\* 此处省略redisTemplate的 setter和getter方法 \*\*\*/
4. @Override
5. **public** void onMessage(Message message, byte[] bytes) {
6. // 获取消息
7. byte[] body = message.getBody();
8. // 使用值序列化器转换
9. String msgBody = (String) getRedisTemplate().getValueSerializer()
10. .deserialize(body);
11. System.err.println(msgBody);
12. // 获取 channel
13. byte[] channel = message.getChannel();
14. // 使用字符串序列化器转换
15. String channelStr = (String) getRedisTemplate().getStringSerializer()
16. .deserialize(channel);
17. System.err.println(channelStr);
18. // 渠道名称转换
19. String bytesStr = **new** String(bytes);
20. System.err.println(bytesStr);
21. }
22. }

3.2.在spring中配置

<bean id="redisMsgListener" class="com.redis.listener.RedisMessageListener">  
    <property name="redisTemplate" ref="redisTemplate"/>  
</bean>

3.3 要给一个监听容器，在 Spring 中已有类 org.springframework.data.redis.listener.RedisMessageListenerContainer。它可以用于监听 Redis 的发布订阅消息，下面的配置就是为了实现这个功能，读者可以通过注释来了解它的配置要点。

*<!--配置redis监听器-->*<bean id="messageListener" class="com.zy.listener.RedisMessageListener">  
 <property name="stringRedisSerializer" ref="stringRedisSerializer"/>  
 <property name="jdkSerializationRedisSerializer" ref="jdkSerializationRedisSerializer"/>  
</bean>  
*<!--配置redis监听器container-->  
<!--配置线程池-->*<bean id="poolTaskScheduler"  
 class="org.springframework.scheduling.concurrent.ThreadPoolTaskScheduler">  
 <property name="poolSize" value="3" />  
</bean>  
*<!--配置容器，topic是监听的通道的名称，自定义的-->*<redis:listener-container connection-factory="connectionFactory" task-executor="poolTaskScheduler" topic-serializer="stringRedisSerializer">  
 <redis:listener ref="messageListener" method="onMessage" topic="chat" serializer="stringRedisSerializer"/>  
</redis:listener-container>

3.4 测试，像chat通道发送消息

1. ApplicationContext applicationContext = **new** ClassPathXmlApplicationContext("applicationContext.xml");
2. RedisTemplate redisTemplate = applicationContext.getBean(RedisTemplate.**class**);
3. String channel = "chat";
4. redisTemplate.convertAndSend(channel, "I am lazy!!");

4 springboot整合redis

4.1新建springboot项目，勾选上redis

4.2配置application.yml

spring:  
 redis:  
 host: 127.0.0.1  
 port: 6379

jedis:  
 pool:  
 max-active: 100  
 max-idle: 10  
 max-wait: 20000  
 lettuce:  
 shutdown-timeout: 1000

4.2 配置redis的配置类

@Configuration  
public class RedisConfig {  
 @Bean  
 public RedisTemplate<String,Object> redisTemplate(RedisConnectionFactory redisConnectionFactory){  
 RedisTemplate<String,Object> redisTemplate = new RedisTemplate<>();  
 redisTemplate.setConnectionFactory(redisConnectionFactory);  
 redisTemplate.setKeySerializer(stringRedisSerializer());  
 redisTemplate.setValueSerializer(jdkSerializationRedisSerializer());  
 return redisTemplate;  
 }  
   
 @Bean  
 public StringRedisSerializer stringRedisSerializer(){  
 return new StringRedisSerializer();  
 }  
 @Bean  
 public JdkSerializationRedisSerializer jdkSerializationRedisSerializer(){  
 return new JdkSerializationRedisSerializer();  
 }  
}

4.3 测试

@SpringBootTest  
class SpringbootRedisApplicationTests {  
@Autowired  
private RedisTemplate<String,Object> redisTemplate;  
 @Test  
 void contextLoads() {  
 redisTemplate.opsForValue().set("myname","zhouyang");  
 String myname = (String)redisTemplate.opsForValue().get("myname");  
 System.*out*.println(myname);  
 }  
  
}

4.4 配置订阅发布

4.4.1新建类实现MessageListener

public class MyRedisMessageListener implements MessageListener {  
 @Autowired  
 private RedisTemplate<String,Object> redisTemplate;  
 @Override  
 public void onMessage(Message message, byte[] bytes) {  
 *// 获取消息* byte[] body = message.getBody();  
 *// 使用值序列化器转换* String msgBody = (String) redisTemplate.getValueSerializer()  
 .deserialize(body);  
 System.*err*.println(msgBody);  
 *// 获取 channel* byte[] channel = message.getChannel();  
 *// 使用字符串序列化器转换* String channelStr = (String) redisTemplate.getStringSerializer()  
 .deserialize(channel);  
 System.*err*.println(channelStr);  
 *// 渠道名称转换* String bytesStr = new String(bytes);  
 System.*err*.println(bytesStr);  
  
 }  
}

4.4.2 配置redisConfig，把messageListener和containner放入容器中

@Bean  
public MyRedisMessageListener myRedisMessageListener(){  
 return new MyRedisMessageListener();  
}  
@Bean  
public RedisMessageListenerContainer redisMessageListenerContainer (RedisConnectionFactory redisConnectionFactory){  
 RedisMessageListenerContainer redisMessageListenerContainer = new RedisMessageListenerContainer();  
 redisMessageListenerContainer.setConnectionFactory(redisConnectionFactory);  
 redisMessageListenerContainer.addMessageListener(myRedisMessageListener(),new PatternTopic("chat"));  
 return redisMessageListenerContainer;  
}

4.4.3测试

@Test  
void testPublish() {  
 redisTemplate.convertAndSend("chat","我是周扬");  
}