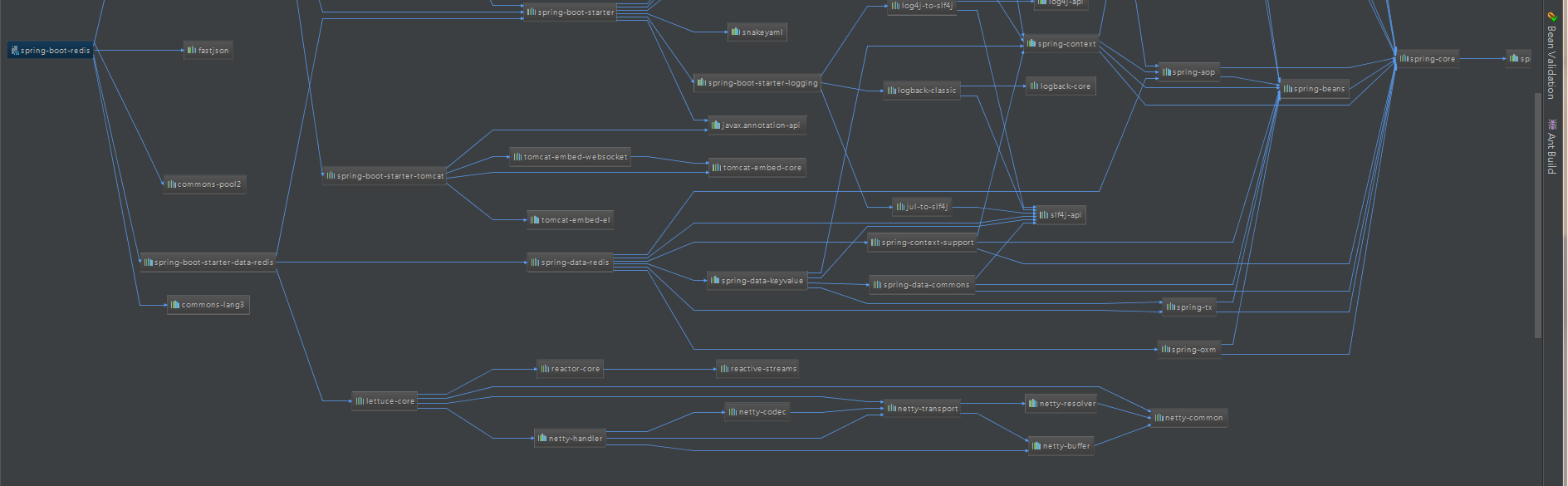
1.导入redis相关jar包。

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-redis</artifactId>  
 <version>${spring-boot.version}</version>  
</dependency>  
<!--spring2.0集成redis所需common-pool2-->  
<!-- spring boot 2.0 的操作手册有标注 大家可以去看看 地址是：https://docs.spring.io/spring-boot/docs/2.0.3.RELEASE/reference/htmlsingle/-->  
<dependency>  
 <groupId>org.apache.commons</groupId>  
 <artifactId>commons-pool2</artifactId>  
 <version>2.5.0</version>  
</dependency>

2.依赖图



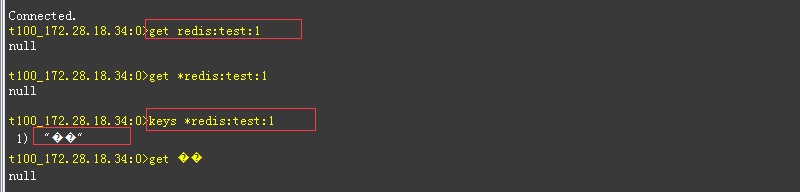
3.application.yml配置

**spring:  
 redis.host:** 172.28.18.34  
 **redis.port:** 6379  
 **redis.timeout:** 1000  
 **redis.password:  
 redis.database:** 0  
 **redis.lettuce.pool.max-active:** 100  
 **redis.lettuce.pool.max-wait:** 100000  
 **redis.lettuce.pool.max-idle:** 8  
 **redis.lettuce.pool.min-idle:** 0

4.代码



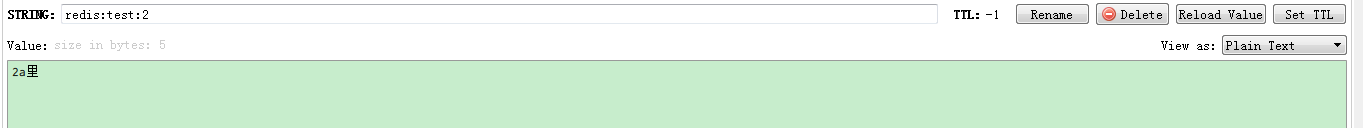
5.客户端查看，你会发现直接输入键值是查询不到结果的，这是因为redisTemplate 的键值序列化默认使用的jdkSerializeable, 存储二进制字节码，所以客户端不易直接查看。



6.为了使客户端便于查询缓存数据，我们可以使用StringRedisTemplate模板对象进行数据存取操作，这样键值的存取都必须是string类型才行。如果值不是字符串就会报错。这种策略通常可以满足我们大部分场景的需要了。

注意不管是键还是值，都只是序列化策略不同而已，跟可以存取和读取的数据结构无关，依然都可以操作String，List，Set，Map等结构数据。

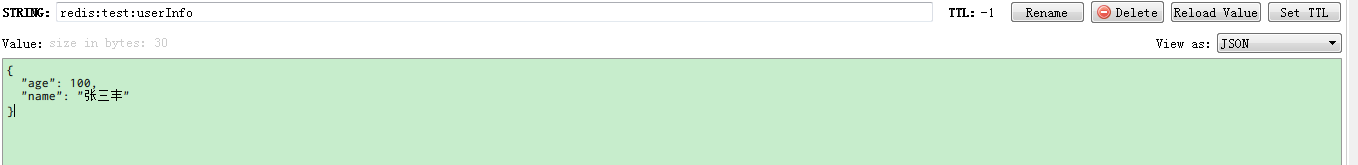
Redis存储的键值都是二进制的，不同的序列化策略本质是在转化为二进制之前的策略不同。字符串的策略其实就是键和值都分别各自必须先是Sting字符串，然后字符串得到对应的btye[]进行磁盘存储的而已。JDK默认的序列化是对象序列化，要多出有额外的一堆描述信息等二进制数据。



7.如果键值的存取都采用String类型，还想使用RedisTemplate对象模板，也可以做到，只要在配置类中重新定制自己的序列化策略即可。

@Configuration  
public class RedisConfiguration implements WebMvcConfigurer {  
  
 */\*\*  
 \* redisTemplate模板对象序列化使用的jdkSerializeable, 存储二进制字节码, 所以自定义序列化类  
 \* 当然,如果键值都是String字符串,其实可以直接使用StringRedisTemplate模板对象进行存取操作了  
 \*  
 \** ***@param*** *redisConnectionFactory  
 \** ***@return*** *\*/* @Bean  
 public RedisTemplate<Object, Object> redisTemplate(RedisConnectionFactory redisConnectionFactory) {  
 RedisTemplate<Object, Object> redisTemplate = new RedisTemplate<>();  
 redisTemplate.setConnectionFactory(redisConnectionFactory);  
  
 // 使用StringRedisSerializer替换默认序列化  
 StringRedisSerializer stringRedisSerializer = new StringRedisSerializer(StandardCharsets.*UTF\_8*);  
  
 // 设置key的序列化规则和value的序列化规则都为String字符串,这样键值的存取都必须是字符串,否则运行时会报错  
 redisTemplate.setKeySerializer(stringRedisSerializer);  
 redisTemplate.setValueSerializer(stringRedisSerializer);  
 redisTemplate.afterPropertiesSet();  
  
 return redisTemplate;  
 }  
  
}

UserInfo userInfo = new UserInfo();  
userInfo.setName("张三丰");  
userInfo.setAge(100);  
  
redisTemplate.opsForValue().set("redis:test:userInfo", JSONObject.*toJSONString*(userInfo));  
Object o = (String)redisTemplate.opsForValue().get("redis:test:userInfo");  
String userInfoStr = null == o ? null : String.*valueOf*(o);  
System.*out*.println(userInfoStr);



8.键值的存取都是String类型可以满足我们的需要，但总是感觉还是不够方便，其实我们最常见，最方便的是键是String类型的，值对于我们来说是“对象”类型的，我们在调用RedisTemplate对象模板的时候最好不要传入字符串，而是直接放入一个对象，但是在往redis进行存储时，

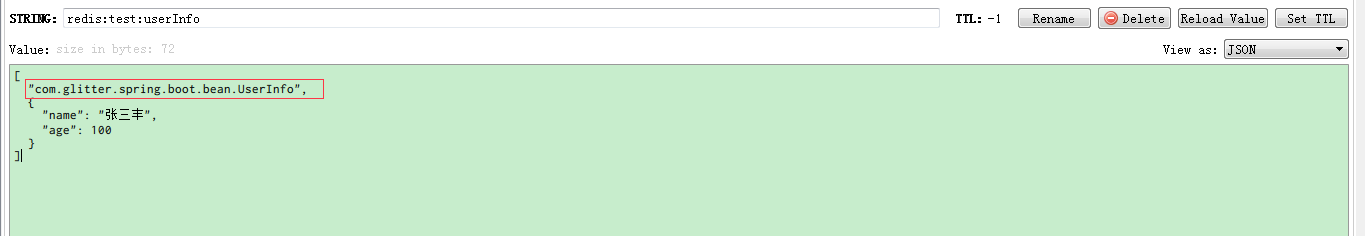
却是先自动进行json化，同时放入对象的字符串描述信息以方便json自动反序列化成对象，然后将json化后的String字符串得到byte[]二进制数据进行磁盘存储。

这样与StringRedisTemplate相比，方便之处在于我们在调用RedisTemplate的api时，不用每次都将对象自己转json串了，直接放入一个对象，读取的时候也不用自己将json字符串反序列化成对象，RedisTemplate的方法会帮我们来转，这样看上去更面向对象一些，同时也确实方便了一点点，何乐而不为。

*/\*\*  
 \* redisTemplate 序列化使用的jdkSerializeable, 存储二进制字节码, 所以自定义序列化类  
 \* 键采用String类型,值采用对象类型,最终是String的Json串进行二进制化存储.  
 \** ***@param*** *redisConnectionFactory  
 \** ***@return*** *\*/*@Bean  
public RedisTemplate<Object, Object> redisTemplate(RedisConnectionFactory redisConnectionFactory) {  
 RedisTemplate<Object, Object> redisTemplate = new RedisTemplate<>();  
 redisTemplate.setConnectionFactory(redisConnectionFactory);  
  
 // 使用Jackson2JsonRedisSerialize替换默认序列化  
 Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new Jackson2JsonRedisSerializer(Object.class);  
  
 ObjectMapper objectMapper = new ObjectMapper();  
 objectMapper.setVisibility(PropertyAccessor.*ALL*, JsonAutoDetect.Visibility.*ANY*);  
 objectMapper.enableDefaultTyping(ObjectMapper.DefaultTyping.*NON\_FINAL*);  
  
 jackson2JsonRedisSerializer.setObjectMapper(objectMapper);  
  
 // 设置value的序列化规则和 key的序列化规则  
 redisTemplate.setKeySerializer(new StringRedisSerializer());  
 redisTemplate.setValueSerializer(jackson2JsonRedisSerializer);  
 redisTemplate.afterPropertiesSet();  
 return redisTemplate;  
}

UserInfo userInfo = new UserInfo();  
userInfo.setName("张三丰");  
userInfo.setAge(100);

redisTemplate.opsForValue().set("redis:test:userInfo", userInfo);  
UserInfo userInfo1 = (UserInfo)redisTemplate.opsForValue().get("redis:test:userInfo");  
System.*out*.println(userInfo1);



9.RedisTemplate模板对象用于操作redis数据还是比较“高层次”的一种操作，Spring在这个工具中为我们封装了很多东西，必须序列化策略等。那么Spring也为我们提供了更“底层”的操作redis的api供我们使用，这就是RedisConnection对象，现在还没有看到使用场景，先记录下用法。

// redisTemplate.execute((RedisCallback<Void>) (RedisConnection con) -> {  
// String key = "redis:test:userInfo";  
// String value = JSONObject.toJSONString(userInfo);  
// con.set(key.getBytes(), value.getBytes());  
// return null;  
// });  
  
// String key = "redis:test:userInfo";  
// String asn = (String)redisTemplate.execute((RedisCallback<String>) con -> con.get(key.getBytes()) == null ? null : new String(con.get(key.getBytes())  
// ));  
// System.out.println(asn);