自定义模板解决序列化问题

默认情况下的模板 RedisTemplate<Object, Object>，默认序列化使用的是`JdkSerializationRedisSerializer`，存储二进制字节码。这时需要自定义模板，当自定义模板后又想存储 String 字符串时，可以使StringRedisTemplate的方式，他们俩并不冲突。

序列化问题：

要把 domain object 做为 key-value 对保存在 redis 中，就必须要解决对象的序列化问题。Spring Data Redis给我们提供了一些现成的方案：

`JdkSerializationRedisSerializer`使用JDK提供的序列化功能。 优点是反序列化时不需要提供类型信息(class)，但缺点是序列化后的结果非常庞大，是JSON格式的5倍左右，这样就会消耗 Redis 服务器的大量内存。

`Jackson2JsonRedisSerializer`使用 Jackson 库将对象序列化为JSON字符串。优点是速度快，序列化后的字符串短小精悍。但缺点也非常致命，那就是此类的构造函数中有一个类型参数，必须提供要序列化对象的类型信息(.class 对象)。通过查看源代码，发现其只在反序列化过程中用到了类型信息。

`GenericJackson2JsonRedisSerializer`通用型序列化，这种序列化方式不用自己手动指定对象的 Class。

```
@Configuration
public class RedisConfig {
    @Bean
    public RedisTemplate<String,Object> redisTemplate(LettuceConnectionFactory
redisConnectionFactory){
        RedisTemplate<String,Object> redisTemplate = new RedisTemplate<>();
        //为string类型key设置序列器
        redisTemplate.setKeySerializer(new StringRedisSerializer());
        //为string类型value设置序列器
        redisTemplate.setValueSerializer(new GenericJackson2JsonRedisSerializer());
        //为hash类型key设置序列器
        redisTemplate.setHashKeySerializer(new StringRedisSerializer());
        //为hash类型value设置序列器
        redisTemplate.setHashValueSerializer(new GenericJackson2JsonRedisSerializer());
        redisTemplate.setConnectionFactory(redisConnectionFactory);
        return redisTemplate;
    }
}
```

```
//序列化
@Test
public void testSerial(){
    User user = new User();
    user.setId(1);
    user.setUsername("张三");
    user.setPassword("111");
    ValueOperations<String, Object> value = redisTemplate.opsForValue();
    value.set("userInfo",user);
    System.out.println(value.get("userInfo"));
}
```