具体如何在项目中操作数据，请看项目代码。最主要的看子模块 redis 的 springbootredis 相关代码，例子展示了如何从 redis 存取基本数据类型和对象。

第一步：导入依赖

```
<!--redis 依赖-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

第二步：配置 redis

创建一个 redis 配置类（本例配置类名称为 RedisConfig），具体代码在最后。

该类的作用：将存到 redis 的数据序列化。

如果不配置这个类，使用“redis desktop manager”查看 redis 缓存的数据时，数据会乱码（想看效果的话，可以试着将本类的代码都注释了，做个对比）

第三步：引入 RedisTemplate

在 serviceImpl 层引入，通过 RedisTemplate 对 redis 进行操作

```
@Autowired
private RedisTemplate redisTemplate;
```

第四步：直接使用 redisTeplate 即可

附：RedisConfig.class 代码

```
@Configuration
@EnableCaching
public class RedisConfig extends CachingConfigurerSupport {
    //自定义缓存 key 生成策略
    @Bean
    public KeyGenerator keyGenerator() {
        return new KeyGenerator(){
            @Override
            public Object generate(Object target, java.lang.reflect.Method method,
Object... params) {
                StringBuffer sb = new StringBuffer();
                sb.append(target.getClass().getName());
                sb.append(method.getName());
                for(Object obj:params){
                    sb.append(obj.toString());
                }
                return sb.toString();
```

```
            }
        };
    }
    @Bean
    public CacheManager cacheManager(RedisConnectionFactory connectionFactory) {
// 基于 java 代码实现数据库查询同步缓存到 reids，不会出现乱码，的
//cachemanager 的配置
        RedisCacheManager redisCacheManager =
RedisCacheManager.create(connectionFactory);
        return  redisCacheManager;
    }
    @Bean
    public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory
factory){
// 基于 java 代码的查询数据库同步更新缓存的代码，不会让存到缓存的数据出
// 现乱码。可用
        RedisTemplate<String, Object> template = new RedisTemplate<>();
// 配置连接工厂
        template.setConnectionFactory(factory);
//使用 Jackson2JsonRedisSerializer 来序列化和反序列化 redis 的 value 值
（默认使用 JDK 的序列化方式）
        Jackson2JsonRedisSerializer jacksonSeial = new
Jackson2JsonRedisSerializer(Object.class);

        ObjectMapper om = new ObjectMapper();
// 指定要序列化的域，field,get 和 set,以及修饰符范围，ANY 是都有包括
//private 和 public
        om.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
// 指定序列化输入的类型，类必须是非 final 修饰的，final 修饰的类，比如
//String,Integer 等会跑出异常
        om.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);
        jacksonSeial.setObjectMapper(om);
         // 值采用 json 序列化
        template.setValueSerializer(jacksonSeial);
        //使用 StringRedisSerializer 来序列化和反序列化 redis 的 key 值
        template.setKeySerializer(new StringRedisSerializer());    //原
        // 设置 hash key 和 value 序列化模式
        template.setHashKeySerializer(new StringRedisSerializer());
        template.setHashValueSerializer(jacksonSeial);
        template.afterPropertiesSet();
        return template;
    }
}
```