

```
//操作redis中字符串 opsForValue 实际操作就是redis中String类型
@Test
public void testString() {
    stringRedisTemplate.opsForValue().set("name", "小陈");
    String value= stringRedisTemplate.opsForValue().get("name");
    System.out.println("value = " + value);
```

```
//注入StringRedisTemplate
@Autowired
private StringRedisTemplate stringRedisTemplate; //key value 都是字符串

//操作redis中key相关

@Test
public void testKey(){
    //stringRedisTemplate.delete("name");//删除一个key
    Boolean hasKey = stringRedisTemplate.hasKey("name");//判断某个key是否存在
    System.out.println(hasKey);
    DataType name = stringRedisTemplate.type(key: "name");//判断key所对应值的类型
    System.out.println(name);
    Set<String> keys = stringRedisTemplate.keys(pattern: "*");//获取redis中所有可以
    keys.forEach(key -> System.out.println("key = " + key));
}

}
```

```
@Test
public void testKey(){
    //stringRedisTemplate.delete("name");//删除一个key
    Boolean hasKey = stringRedisTemplate.hasKey("name");//判断某个key是否存在
    System.out.println(hasKey);

    DataType name = stringRedisTemplate.type(key: "name");//判断key所对应值的类型
    System.out.println(name);

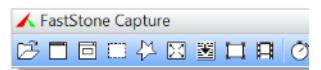
    Set<String> keys = stringRedisTemplate.keys(pattern: "*");//获取redis中所有key
    keys.forEach(key -> System.out.println("key = " + key));

    Long expire = stringRedisTemplate.getExpire(key: "age");//获取key超时时间 -1 永不超时 -2 key不存在 >=0 过期时间
    System.out.println(expire);

    stringRedisTemplate.randomKey();//在redis中随机获取一个key

    //stringRedisTemplate.rename("age", "age1");//修改key名字 要求key必须存在 不存在 报错
    //stringRedisTemplate.renameIfAbsent("name", "name1");//修改key名字 判断key是否存在

    stringRedisTemplate.move(key: "name1", dbIndex: 1);//移动key
```



```
//操作redis中字符串 opsForValue 实际操作就是redis中String类型
@Test
public void testString(){
    stringRedisTemplate.opsForValue().set("name", "小陈"); //set 用来设置一个key value

    String value= stringRedisTemplate.opsForValue().get("name"); //用来获取一个key对应value
    System.out.println("value = " + value);

    stringRedisTemplate.opsForValue().set( k: "code", v: "2357", l: 120, TimeUnit.SECONDS); //设置一个key 超时时间

    stringRedisTemplate.opsForValue().append( k: "name", s: "他是是一个好人,单纯少年!"); //追加
}
```





```
//操作redis中list类型    opsForList 实际操作就是redis中list类型
@Test
public void testList(){
    //stringRedisTemplate.opsForList().leftPush("names", "小陈");//创建一个列表 并放入一个元素
    //stringRedisTemplate.opsForList().leftPushAll("names", "小陈", "小张", "小王");//创建一个列表 放入多个元素
    List<String> names = new ArrayList<>();
    names.add("xiaoming");
    names.add("xiaosan");
    //stringRedisTemplate.opsForList().leftPushAll("names", names);//创建一个列表 放入多个元素

    List<String> stringList = stringRedisTemplate.opsForList().range( k: "names", |: 0, |1: -1); //遍历list
    stringList.forEach(value-> System.out.println("value = " + value));

    stringRedisTemplate.opsForList().trim( k: "names", |: 1, |1: 3); //截取指定区间的list
}
```

```
@Test  
public void testSet(){  
    stringRedisTemplate.opsForSet().add( k: "sets", ...vs: "张三", "张三", "小陈", "xiaoming"); //创建set 并放入多个元素  
  
    Set<String> sets = stringRedisTemplate.opsForSet().members( k: "sets"); //查看set中成员  
    sets.forEach(value-> System.out.println("value = " + value));  
  
    Long size = stringRedisTemplate.opsForSet().size( k: "sets"); //获取set集合元素个数  
    System.out.println("size = " + size);  
}
```



```
//操作redis中Zset类型    opsForSet 实际操作就是redis中set类型
@Test
public void testZset() {
    stringRedisTemplate.opsForZSet().add( k: "zsets", v: "小黑", v1: 20); //创建并放入元素

    Set<String> zsets = stringRedisTemplate.opsForZSet().range( k: "zsets", l: 0, l1: -1); //指定范围查询
    zsets.forEach(value-> System.out.println(value));
    System.out.println("====");
    Set<ZSetOperations.TypedTuple<String>> zsets1 = stringRedisTemplate.opsForZSet().rangeByScoreWithScores( k: "z"
    zsets1.forEach(typedTuple ->{
        System.out.println(typedTuple.getValue());
        System.out.println(typedTuple.getScore());
    });
}
```



```
//操作redis中Hash类型    opsForHash 实际操作就是redis中Hash类型

@Test
public void testHash(){

    stringRedisTemplate.opsForHash().put( h: "maps", hk: "name", hv: "张三"); //创建一个hash类型 并放入key value

    Map<String, String> map = new HashMap<String, String>();
    map.put("age", "12");
    map.put("bir", "2012-12-12");
    stringRedisTemplate.opsForHash().putAll( h: "maps", map); //放入多个key value

    List<Object> values = stringRedisTemplate.opsForHash().multiGet( h: "maps", Arrays.asList("name", "age")); //获取
    values.forEach(value-> System.out.println(value));

    String value = (String) stringRedisTemplate.opsForHash().get( h: "maps", o: "name"); //获取hash中某个key的值

    List<Object> vals = stringRedisTemplate.opsForHash().values( h: "maps"); //获取所有values

    Set<Object> keys = stringRedisTemplate.opsForHash().keys( h: "maps"); //获取所有keys

}
```



```
public void testRedisTemplate() {  
  
    /**  
     * redisTemplate对象中 key 和 value 的序列化都是 JdkSerializationRedisSerializer  
     *      key: string  
     *      value: object  
     *      修改默认key序列化方案 : key StringRedisSerializer  
     */  
  
    //修改key序列化方案 String类型序列  
    redisTemplate.setKeySerializer(new StringRedisSerializer());  
    //修改hash key 序列化方案  
    redisTemplate.setHashKeySerializer(new StringRedisSerializer());  
  
    User user = new User();  
    user.setId(UUID.randomUUID().toString()).setName("小陈").setAge(23).setBir(new Date());  
    redisTemplate.opsForValue().set("user", user); //redis进行设置 对象需要经过序列化  
  
    User user1 = (User) redisTemplate.opsForValue().get("user");  
    System.out.println(user1);  
  
    redisTemplate.opsForList().leftPush("list", user);  
  
    redisTemplate.opsForSet().add(key: "set", user);  
  
    redisTemplate.opsForZSet().add(key: "zset", user, score: 10);  
  
    redisTemplate.opsForHash().put(key: "map", hashKey: "name", user);  
  
}
```

```
//spring data 为了方便我们对redis进行更友好的操作 因此有提供了bound api 简化操作
@Test
public void testBound() {

    redisTemplate.setKeySerializer(new StringRedisSerializer());
    redisTemplate.setHashKeySerializer(new StringRedisSerializer());

    //redisTemplate  stringRedisTemplate 将一个key多次操作进行绑定 对key绑定
    stringRedisTemplate.opsForValue().set("name", "zhangsan");
    stringRedisTemplate.opsForValue().append("name", "是一个好人");
    String s = stringRedisTemplate.opsForValue().get("name");
    System.out.println(s);

    //对字符串类型key进行绑定 后续所有操作都是基于这个key的操作
    BoundValueOperations<String, String> nameValueOperations = stringRedisTemplate.boundValueOps(key: "name");
    nameValueOperations.set("zhangsan");
    nameValueOperations.append("是一个好人");
    String s1 = nameValueOperations.get();
    System.out.println(s1);
}
```

```
/***
 * redis应用场景
 *
 * 1.利用redis 中字符串类型完成 项目中手机验证码存储的实现
 * 2.利用redis 中字符串类型完成 具有失效性业务功能 12306 淘宝 订单还有:40分钟
 * 3.利用redis 分布式集群系统中 Session共享 memcache 内存 数据存储上限 数据类型比较简单 redis 内存 数据上限 数据类型丰富
 * 4.利用redis zset类型 可排序set类型 元素 分数 排行榜之类功能 dangdang 销量排行 sales(zset) [商品id,商品销量] .....
 * 5.利用redis 分布式缓存 实现
 * 6.利用redis 存储认证之后token信息 微信小程序 微信公众号 |用户 openid ---> 令牌(token) redis 超时
 * 7.利用redis 解决分布式集群系统中分布式锁问题 redis 单进程 单线程 n 20 定义
 *      jvm 1进程开启多个线程 synchronize int n=20
 *      jvm 1进程开启多个线程 synchronize int n=20
 *      ..... LRA脚本
 */

```