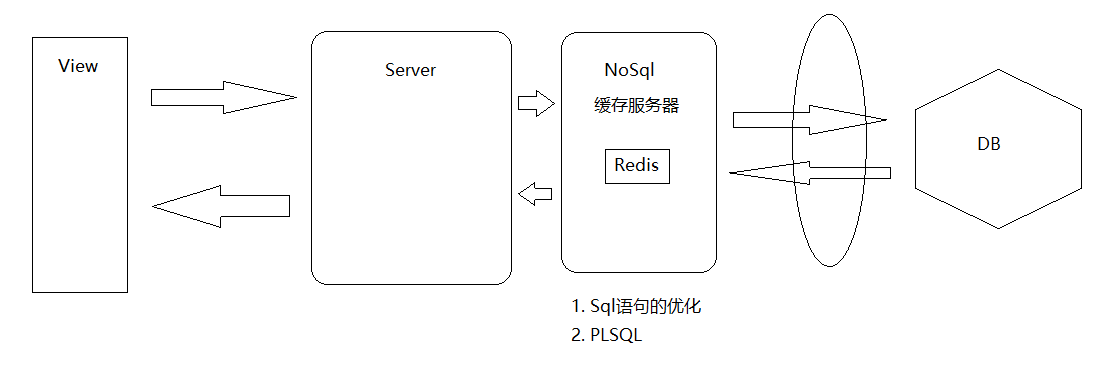
# NoSql

## 基本

NoSQL，泛指非关系型的数据库。随着互联网web2.0网站的兴起，传统的关系数据库在应付web2.0网站，特别是超大规模和高并发的SNS类型的web2.0纯动态网站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。NoSQL数据库的产生就是为了解决大规模数据集合多重数据种类带来的挑战，尤其是大数据应用难题。



## Redis

Redis是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的API

redis是一个key-value存储系统。

安装

redis服务器(windows版本)

Redis-x64-3.2.100.msi

redis的可视化的视图操作工具

redis-desktop-manager-0.8.8.384.exe

# Redis操作

## Redis-cli

redis的客户端连接指令.定位到redis的根目录



注意:

* redis服务器的默认端口为6379

如果需要连接远程的redis的服务器

redis-cli -h ip地址 -p 6379

## Redis的五种数据类型

### String(key-value)

Redis 字符串数据类型的相关命令用于管理 redis 字符串值.采用key-value的形式,redis中命令不区分大小写,key是区分大小写.

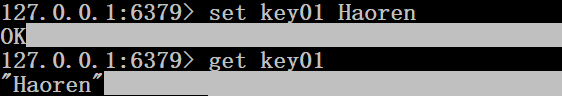
* set 指令设置键值对

set key value

* get 指令根据key获得对应的值

get key

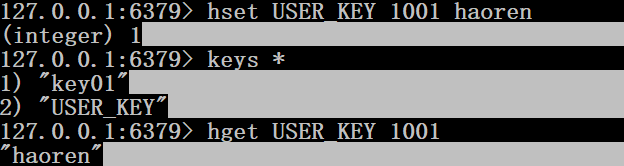
案例:



### Hash (key-fields-values)

Redis hash 是一个string类型的field和value的映射表，hash特别适合用于存储对象。Hash结构通过key进行归类,Redis 中每个 hash 可以存储 232 - 1 键值对（40多亿）。

* hset:向Hash中添加数据
* hget:从Hash中取数据

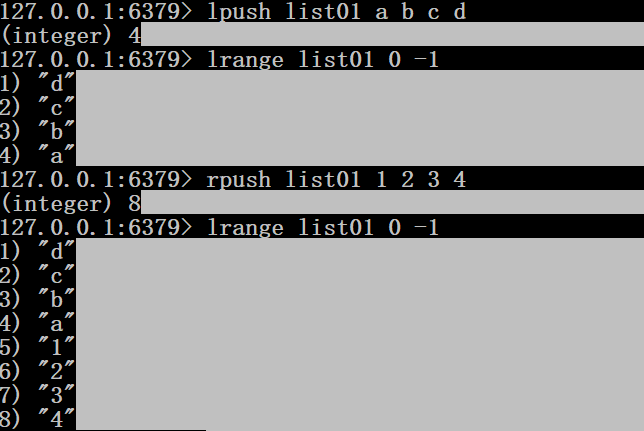


### List

存储有顺序课重复的数据, Redis列表是简单的字符串列表，按照插入顺序排序。你可以添加一个元素到列表的头部（左边）或者尾部（右边）

* LPUSH key value1 [value2] :将一个或多个值插入到列表头部
* RPUSH key value1 [value2] : 在列表中添加一个或多个值
* LRANGE key start stop : 获取列表指定范围内的元素
* LLEN key :获取列表长度
* RPOP key : 移除列表的最后一个元素，返回值为移除的元素。
* LPOP key : 移出并获取列表的第一个元素

案例:

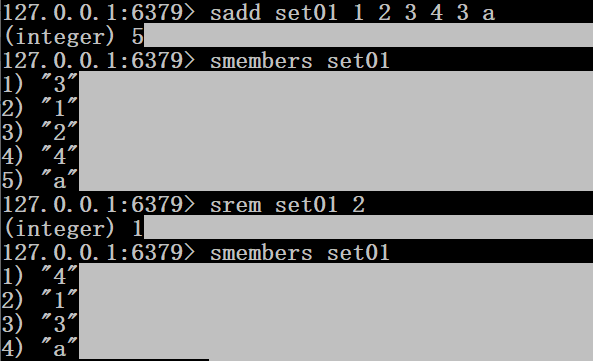


### Set

Redis 的 Set 是 String 类型的无序集合。集合成员是唯一的，这就意味着集合中不能出现重复的数据。

* SADD key member1 [member2] : 向集合添加一个或多个成员
* SMEMBERS key :返回集合中的所有成员
* SISMEMBER key member :判断 member 元素是否是集合 key 的成员
* SREM key member1 [member2] :移除集合中一个或多个成员

案例:



### sorted set

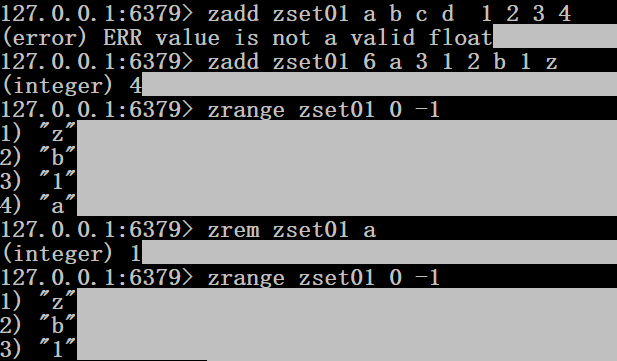
Redis 有序集合和集合一样也是string类型元素的集合,且不允许重复的成员。不同的是每个元素都会关联一个double类型的分数。redis正是通过分数来为集合中的成员进行从小到大的排序。有序集合的成员是唯一的,但分数(score)却可以重复。

* ZADD key score1 member1 [score2 member2] :向有序集合添加一个或多个成员，或者更新已存在成员的分数
* ZRANK key member :返回有序集合中指定成员的索引
* ZREVRANK key member:返回有序集合中指定成员的排名，有序集成员按分数值递减(从大到小)排序

案例:

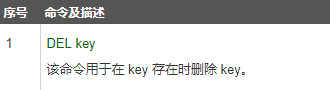
* ZREM key member [member ...] :移除有序集合中的一个或多个成员

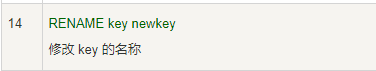
案例:



## key相关的指令

Redis 键命令用于管理 redis 的键。

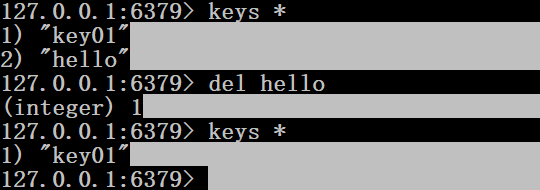








案例



# Jedis

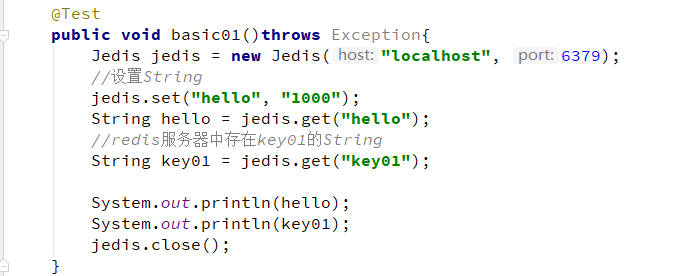
## 基本概念

Jedis Client是Redis官网推荐的一个面向java客户端，库文件实现了对redis各类API进行封装调用。redis通信协议是Redis客户端与Redis Server之间交流的语言，它规定了请求和返回值的格式。redis-cli与server端使用一种专门为redis设计的协议RESP(Redis Serialization Protocol)交互，Resp本身没有指定TCP，但redis上下文只使用TCP连接。

**导入依赖**

<dependency>  
 <groupId>redis.clients</groupId>  
 <artifactId>jedis</artifactId>  
 <version>3.0.1</version>  
</dependency>

### 测试代码



连接池的配置



## 业务逻辑使用redis

提供对于Redis操作的基本接口

public interface JedisClient {

String set(String key, String value);

String get(String key);

Boolean exists(String key);

Long expire(String key, int seconds);

Long ttl(String key);

Long incr(String key);

Long hset(String key, String field, String value);

String hget(String key, String field);

Long hdel(String key, String... field);

}

提供实现类

public class JedisClientPool implements JedisClient {

@Autowired

private JedisPool jedisPool;

@Override

public String set(String key, String value) {

Jedis jedis = jedisPool.getResource();

String result = jedis.set(key, value);

jedis.close();

return result;

}

@Override

public String get(String key) {

Jedis jedis = jedisPool.getResource();

String result = jedis.get(key);

jedis.close();

return result;

}

@Override

public Boolean exists(String key) {

Jedis jedis = jedisPool.getResource();

Boolean result = jedis.exists(key);

jedis.close();

return result;

}

@Override

public Long expire(String key, int seconds) {

Jedis jedis = jedisPool.getResource();

Long result = jedis.expire(key, seconds);

jedis.close();

return result;

}

@Override

public Long ttl(String key) {

Jedis jedis = jedisPool.getResource();

Long result = jedis.ttl(key);

jedis.close();

return result;

}

@Override

public Long incr(String key) {

Jedis jedis = jedisPool.getResource();

Long result = jedis.incr(key);

jedis.close();

return result;

}

@Override

public Long hset(String key, String field, String value) {

Jedis jedis = jedisPool.getResource();

Long result = jedis.hset(key, field, value);

jedis.close();

return result;

}

@Override

public String hget(String key, String field) {

Jedis jedis = jedisPool.getResource();

String result = jedis.hget(key, field);

jedis.close();

return result;

}

@Override

public Long hdel(String key, String... field) {

Jedis jedis = jedisPool.getResource();

Long result = jedis.hdel(key, field);

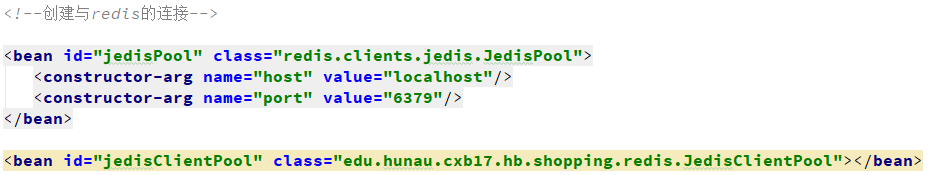
jedis.close();

return result;

}

}

## Spring创建连接



## 业务中追加缓存

1. 先查询缓存
2. 如果存在数据,直接返回
3. 如果不存在数据
4. 查询数据
5. 将数据存储到缓存中

注意: 添加缓存不能影响正常的业务逻辑.

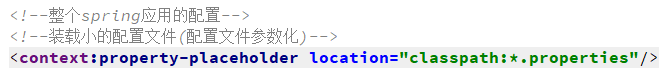


Json的工具类:实现java对象和字符串之间的相互转换.

配置文件中管理key



spring映射文件中



业务层中

