XDL 兄弟连IT教育
www.itxdl.cn

版权归兄弟连所有·2016年版

企业级框架 Redis

Unit10

Redis安装和配置

什么是Redis

- Redis是使用标准C编写实现，而且完全在内存中保存数据
- Redis是一个开源的，先进的 key-value 存储可用于构建高性能，可扩展的 Web 应用程序的解决方案
- Redis可以看做“内存中的数据结构服务器”。目前Redis支持列表、集合、可排序集合、哈希等数据结构。
- Redis支持多种语言，诸如Java 、PHP、Ruby、Python 、Lua等
- Redis官方网网站是：http://www.redis.io/

什么是Redis

- Redis优点
  - 异常快速：每秒可以执行大约110000设置操作，81000个读取操作。
  - 支持丰富的数据类型，例如列表，集合，可排序集合，哈希等数据类型。
  - 支持数据的持久化，Redis提供了一些策略将内存中的数据异步地保存到磁盘上，比如根据时间或更新次数。
  - Redis是一个多功能实用工具，具有缓存、消息传递等功能。

Redis命令

启动Redis客户端

- Redis命令用于在redis服务器上执行某些操作，需要在一个Redis客户端执行。
- 启动Redis客户端访问本机Redis服务器命令如下：

```
$redis -cli
redis 127.0.0.1:6379>
```

启动Redis客户端

- 启动Redis客户端访问远程Redis服务器命令格式：

```
$ redis-cli -h host -p port -a password
```

- 例如

```
$redis-cli -h 127.0.0.1 -p 6379 -a "mypass"
redis 127.0.0.1:6379>
```

字符串操作命令

- 存储字符串

```
redis 127.0.0.1:6379> SET name “tom”
OK
```

- 获取字符串

```
redis 127.0.0.1:6379> GET name
“tom”
```

字符串操作命令-1

字符串操作命令	功能描述
SET key value	此命令用于在指定键设置值
GET key	键对应的值
GETSET key value	设置键的字符串值，并返回旧值
STRLEN key	得到存储在键的值的长度
MSET key value [key value ...]	设置多个键和多个值
INCR key	键的整数值加1

字符串操作命令-2

字符串操作命令	功能描述
INCRBY key value	键的整数值加value
DECR key	键的整数值减1
DECRBY key value	键的整数值减value
APPEND key value	为原来键值追加value

管理Redis键

- Redis的keys命令用于管理键

```
redis 127.0.0.1:6379> COMMAND KEY_NAME
```

- 例如

```
redis 127.0.0.1:6379> SET xdl redis
OK
redis 127.0.0.1:6379> DEL xdl
(integer) 1
```

管理Redis键-1

管理键的命令	功能描述
DEL key	此命令删除键，如果存在
EXISTS key	此命令检查该键是否存在
EXPIRE key seconds	指定键的过期时间
PEXPIRE key milliseconds	设置键以毫秒为单位到期
PERSIST key	移除过期的键
KEYS pattern	查找与指定模式匹配的所有键
DUMP key	该命令返回存储在指定键的值的序列化结果

管理Redis键-2

管理键的命令	功能描述
RANDOMKEY	从Redis返回随机键
RENAME key newkey	更改键的名称
TYPE key	返回存储在键的数据类型的值
PTTL key	以毫秒为单位获取剩余时间的到期键
TTL key	获取键到期的剩余时间

哈希操作命令

- Redis的哈希值是字符串字段和字符串值之间的映射，所以他们是表示对象的完美数据类型
- 它和java的HashMap完全类似
- 在Redis中的哈希值，可存储超过40亿键值对

键名 字段名1 字段值1 字段名2 字段值2 ...

哈希操作命令

- 存储哈希值

```
redis 127.0.0.1:6379> HMSET myhash field1 "foo" field2 "bar"
OK
```

- 获取哈希值

```
redis 127.0.0.1:6379> HGET myhash field1
redis 127.0.0.1:6379> HMGET myhash field2 field2
```

哈希操作命令

哈希操作命令	功能描述
HMSET key field1 value1 [field2 value2]	设置多个哈希字段的多个值
HSET key field value	设置哈希字段的字符串值
HGET key field	获取存储在指定的键散列字段的值
HMGET key field1 [field2]	获得所有给定的哈希字段的值
HLEN key	设置多个键和多个值
HKEYS key	获取所有哈希表中的字段
HDEL key field2 [field2]	删除一个或多个哈希字段
HEXISTS key field	判断一个哈希字段存在与否

列表操作命令

- Redis列表是简单的字符串列表，按插入顺序排序。您可以在头部或列表的尾部给列表添加元素。
- 列表的最大长度为2^32-1个元素
- 它和java的List类似

列表操作命令

- 存储列表值

```
redis 127.0.0.1:6379> LPUSH tutorials redis
(integer) 1
redis 127.0.0.1:6379> LPUSH tutorials mongodb
(integer) 2
```

- 获取列表值

```
redis 127.0.0.1:6379> LRANGE tutorials 0 10
1)  "mongodb"
2)  "redis"
```

列表操作命令-1

列表操作命令	功能描述
LPUSH key value1 [value2]	在前面加上一个或多个值的列表
RPUSH key value1 [value2]	在末尾加上一个或多个值的列表
LRANGE key start stop	返回存储在key列表的特定元素, 0是第一元素(该列表的头部)，1是列表的下一个元素, -1是该列表的最后一个元素，-2倒数第二个
LLEN key	获取列表的长度
LPOP key	从头部删除一个元素，并返回该删除的元素
RPOP key	从尾部删除一个元素，并返回该删除的元素

列表操作命令-2

列表操作命令	功能描述
LTRIM key begin end	对列表元素剪切,保留指定key范围内的数据
LINDEX key index	从一个列表其索引获取对应的元素
LPUSHX key value	在前面加上一个值列表，仅当列表中存在，不成功返回0
RPUSHX key value	在末尾加上一个值列表，仅当列表中存在，不成功返回0
LREM key count value	移除等于value的元素，当count>0时，从表头开始查找，移除count个；当count=0时，从表头开始查找，移除所有等于value的；当count<0时，从表尾开始查找，移除|count| 个
linsert key before|after pivot value	将值插入到pivot的前面或后面。如果有多个pivot，以离表头最近的为准
lset key index value	设置列表指定索引的值，如果索引不存在则报错

集合操作

- Redis的集合是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。它与Java中的Set集合相似
- 集合类型和列表的区别
  - 集合类型和列表类型还是都能存储2^32-1个字符串
  - 集合类型是无序的，列表类型是有序的
  - 集合类型的元素是唯一、不重复的，列表类型的元素允许重复

集合操作

- 存储集合值

```
redis 127.0.0.1:6379> sadd setA 1 2 3
(integer) 3
redis 127.0.0.1:6379> sadd setB 2 3 4
(integer) 3
```

- 获取集合值

```
redis 127.0.0.1:6379> smembers setA
1)  “1"
2)  “2"
3)  “3”
```

集合操作命令

集合操作命令	功能描述
sadd key member [member ...]	向集合增加元素
srem key member [member ...]	从集合删除元素
smembers key	获得集合中的所有元素
sismember key member	判断元素是否在集合中
scard key	获得集合中元素的个数
srandmember key [count]	随机获得集合中的元素
spop key	从列表中弹出一个元素，弹出元素被删除，不在原来的集合中

集合操作命令

集合操作命令	功能描述
sdiff key [key ...]	执行差集运算。集合A与集合B的差值表示为A-B
sinter key [key ...]	执行交集运算。集合A与集合B的交集表示为A∩B
sunion key [key ...]	执行并集运算。集合A与集合B的并集
smove key1 key2 value	将第一个集合元素value移动到第二个集合中
sdiffstore key key1 key2	执行差集运算并且存储到另一个set中
sinterstore key key1 key2	执行交集运算并且存储到另一个set中
sunionstore key key1 key2	执行并集运算并且存储到另一个set中

有序集合操作

- 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。

有序集合操作

- 应用A：比如一个存储全班同学成绩的Sorted Sets，其集合value可以是同学的学号，而score就可以是其考试得分，这样在数据插入集合的时候，就已经进行了天然的排序。
- 应用B：比如普通消息的score为1，重要消息的score为2，然后工作线程可以选择按score的倒序来获取工作任务。这样就可以让重要的任务优先执行。

有序集合操作命令

有序集合操作命令	功能描述
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
ZCARD key	获取有序集合的成员数
ZCOUNT key min max	计算在有序集合中指定区间分数的成员数
ZSCORE key member	获得元素的分数
ZRANGE key start stop [WITHSCORES]	通过索引区间返回有序集合成指定区间内的成员(小到大)
ZREVRANGE key start stop [WITHSCORES]	通过索引区间返回有序集合成指定区间内的成员(大到小)
ZRANGEBYSCORE key score1 score2	根据排序索引的scores来返回元素

有序集合操作命令

有序集合操作命令	功能描述
ZRANK key member	返回元素在集合中的排序位置，就是索引值
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
ZREM key member	删除名称为key的zset中的元素member
ZINCRBY key number member	可以增加一个元素的分数，返回值是更改后的分数
ZREMRANGEBYRANK key start stop	按照元素分数从小到大顺序删除指定范围内所有的元素
ZREMRANGEBYSCORE key min max	删除集合中在给定排序区间的元素 (按score删除)

Redis事务操作命令

- 事务应用

```
redis 127.0.0.1:6379> MULTI
OK
redis 127.0.0.1:6379> SET book-name "C++ in 21 days"
QUEUED
redis 127.0.0.1:6379> GET book-name
QUEUED
redis 127.0.0.1:6379> SADD tag "C++" "Programming"
"Mastering Series"
QUEUED
redis 127.0.0.1:6379> EXEC
```

Redis事务操作命令

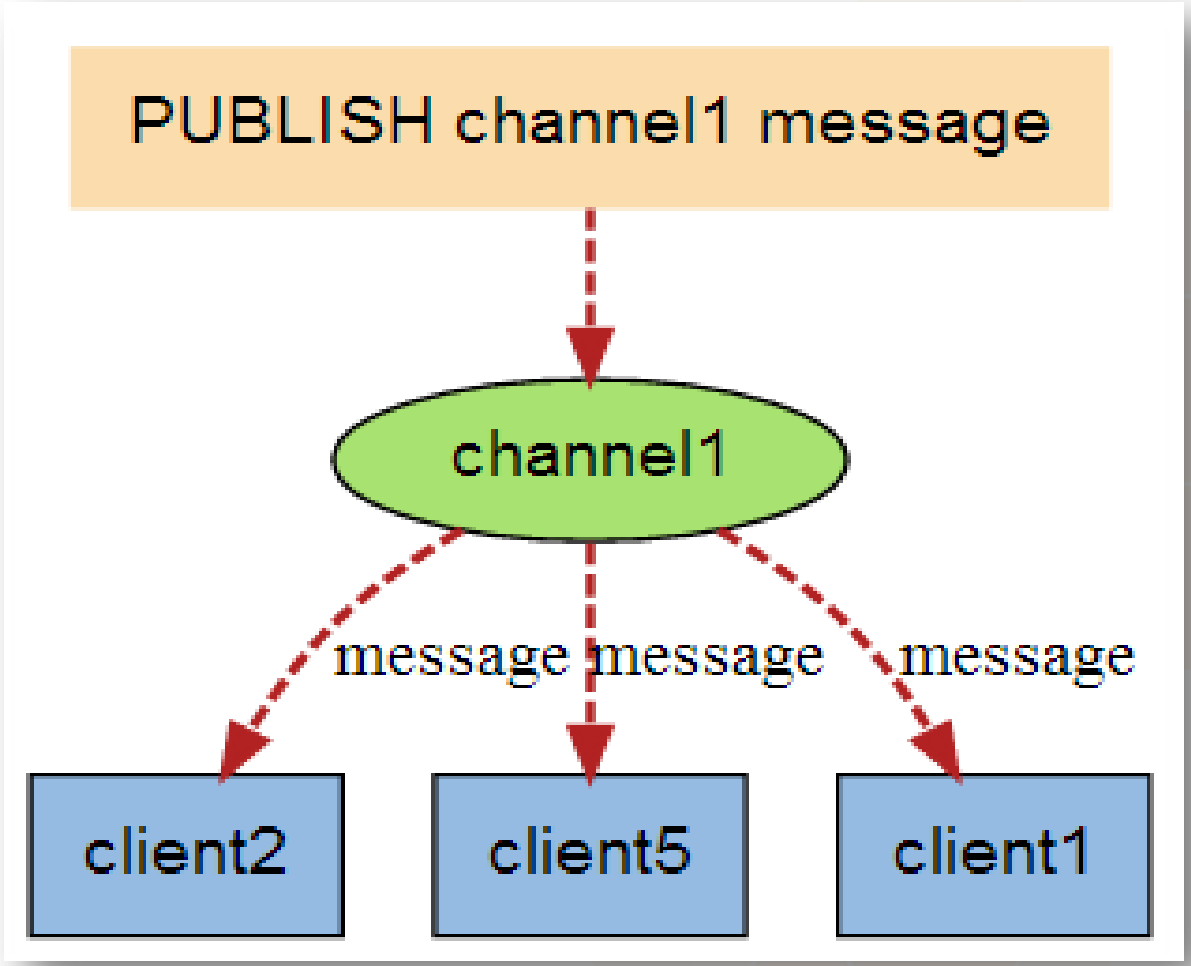
事务操作命令	功能描述
MULTI	标记一个事务块的开始
EXEC	执行所有事务块内的命令
DISCARD	取消事务，放弃执行事务块内的所有命令

Redis备份和恢复命令

事务操作命令	功能描述
SAVE	该命令将在 redis 安装目录中创建dump.rdb文件
BGSAVE	创建 redis 备份文件也可以使用命令 BGSAVE，该命令在后台执行
CONFIG GET dir	获取 redis 安装目录。数据恢复时只需要将导出的dump.rdb文件移动到 redis 安装目录并启动服务即可

```
redis 127.0.0.1:6379> SAVE
OK
```

Redis发布/订阅服务



Redis发布/订阅服务

- 创建了订阅频道名为 redisChat

```
rtedis 127.0.0.1:6379> SUBSCRIBE redisCha
```

- 重新开启个 redis 客户端，然后在同一个频道 redisChat 发布两次消息，订阅者就能接收到消息

```
redis 127.0.0.1:6379> PUBLISH redisChat
 “hello “
redis 127.0.0.1:6379> PUBLISH redisChat  “扣你起哇"
```

Redis发布/订阅服务

事务操作命令	功能描述
SUBSCRIBE channel [channel ...]	订阅给定的一个或多个频道的信息
PUBLISH channel message	将信息发送到指定的频道
UNSUBSCRIBE [channel [channel ...]]	指退订给定的频道

Java使用Redis

Java使用Redis

- Java访问Redis的主要步骤如下：
  - 首先需要下载驱动包，下载 jedis.jar
  - 为工程添加jar包引入

Java使用Redis

- Java访问Redis示例代码如下

```
import redis.clients.jedis.Jedis;
public class RedisJava {
    public static void main(String[] args) {
        //连接本地的 Redis 服务
        Jedis jedis = new Jedis("localhost");
        System.out.println("Connection to server sucessfully");
        //查看服务是否运行
        System.out.println("Server is running: "+jedis.ping());
    }
}
```

Java使用Redis

- Java访问Redis示例代码如下（基于连接池）

```
JedisPoolConfig config = new JedisPoolConfig();

config.setMaxTotal(1024); // 可用连接实例的最大数目,如果赋值为-1,表示不限制.
config.setMaxIdle(5); // 控制一个Pool最多有多少个状态为idle(空闲的)jedis实例,默认值也是8
config.setMaxWaitMillis(1000 * 100); // 等待可用连接的最大时间,单位毫秒,默认值为-1,表示永不超时/如果超过等待时间,则直接抛出异常
jedisPool = new JedisPool(config, "127.0.0.1", 6379);
```

Java使用Redis

- 字符串操作的示例代码如下

```
public class RedisStringJava {
   public static void main(String[] args) {
         //连接本地的 Redis 服务
         Jedis jedis = new Jedis("localhost");
          //设置 redis 字符串数据
          jedis.set("runoobkey", "Redis tutorial");
          // 获取存储的数据并输出
          System.out.println("Stored string in redis: "+
                  jedis.get("runoobkey"));
   }
}
```

Java使用Redis

- List(列表) 操作示例代码如下

```
public static void main(String[] args) {
    //连接本地的 Redis 服务
    Jedis jedis = new Jedis("localhost");
    //存储数据到列表中
    jedis.lpush("tutorial-list", "Redis");
    jedis.lpush("tutorial-list", "Mongodb");
    jedis.lpush("tutorial-list", "Mysql");
    // 获取存储的数据并输出
    List<String> list = jedis.lrange("tutorial-list", 0 ,5);
    for(int i=0; i<list.size(); i++) {
        System.out.println("Stored string in redis:
"+list.get(i));
    }
}
```

Java使用Redis

- Keys操作的示例代码如下

```
public static void main(String[] args) {
    //连接本地的 Redis 服务
    Jedis jedis = new Jedis("localhost");
    System.out.println("Connection to server
sucessfully");
    // 获取数据并输出
    List<String> list = jedis.keys("*");
    for(int i=0; i<list.size(); i++) {
        System.out.println("List of stored keys:
"+list.get(i));
    }
}
```

总结和答疑

XDL 兄弟连IT教育
www.itxdl.cn

版权归兄弟连所有·2016年版