

# Redis

1. 概念： redis是一款高性能的NOSQL系列的非关系型数据库

## 1.1.什么是NOSQL

NoSQL(NoSQL = Not Only SQL), 意即“不仅仅是SQL”, 是一项全新的数据库理念, 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。

### 1.1.1. NOSQL和关系型数据库比较

优点:

- 1) 成本: nosql数据库简单易部署, 基本都是开源软件, 不需要像使用oracle那样花费大量成本购买使用, 相比关系型数据库价格便宜。
- 2) 查询速度: nosql数据库将数据存储于缓存之中, 关系型数据库将数据存储于硬盘中, 自然查询速度远不及nosql数据库。
- 3) 存储数据的格式: nosql的存储格式是key,value形式、文档形式、图片形式等等, 所以可以存储基础类型以及对象或者是集合等各种格式, 而数据库则只支持基础类型。
- 4) 扩展性: 关系型数据库有类似join这样的多表查询机制的限制导致扩展很艰难。

缺点:

- 1) 维护的工具和资料有限, 因为nosql是属于新的技术, 不能和关系型数据库10几年的技术同日而语。
- 2) 不提供对sql的支持, 如果不支持sql这样的工业标准, 将产生一定用户的学习和使用成本。
- 3) 不提供关系型数据库对事务的处理。

### 1.1.2. 非关系型数据库的优势:

- 1) 性能NOSQL是基于键值对的, 可以想象成表中的主键和值的对应关系, 而且不需要经过SQL层的解析, 所以性能非常高。
- 2) 可扩展性同样也是因为基于键值对, 数据之间没有耦合性, 所以非常容易水平扩展。

### 1.1.3. 关系型数据库的优势:

- 1) 复杂查询可以用SQL语句方便的在一个表以及多个表之间做非常复杂的数据查询。
- 2) 事务支持使得对于安全性能很高的数据访问要求得以实现。对于这两类数据库, 对方的优势就是自己的弱势, 反之亦然。

### 1.1.4. 总结

关系型数据库与NoSQL数据库并非对立而是互补的关系, 即通常情况下使用关系型数据库, 在适合使用NoSQL的时候使用NoSQL数据库,

让NoSQL数据库对关系型数据库的不足进行弥补。

一般会将数据存储于关系型数据库中, 在nosql数据库中备份存储关系型数据库的数据

## 1.2.主流的NOSQL产品

- 键值(Key-Value)存储数据库

相关产品: Tokyo Cabinet/Tyrant、Redis、Voldemort、Berkeley DB

典型应用: 内容缓存, 主要用于处理大量数据的高访问负载。

数据模型：一系列键值对

优势：快速查询

劣势：存储的数据缺少结构化

- 列存储数据库

相关产品：Cassandra, HBase, Riak

典型应用：分布式的文件系统

数据模型：以列簇式存储，将同一列数据存在一起

优势：查找速度快，可扩展性强，更容易进行分布式扩展

劣势：功能相对局限

- 文档型数据库

相关产品：CouchDB、MongoDB

典型应用：Web应用（与Key-Value类似，Value是结构化的）

数据模型：一系列键值对

优势：数据结构要求不严格

劣势：查询性能不高，而且缺乏统一的查询语法

- 图形(Graph)数据库

相关数据库：Neo4J、InfoGrid、Infinite Graph

典型应用：社交网络

数据模型：图结构

优势：利用图结构相关算法。

劣势：需要对整个图做计算才能得出结果，不容易做分布式的集群方案。

### 1.3 什么是Redis

Redis是用C语言开发的一个开源的高性能键值对（key-value）数据库，官方提供测试数据，50个并发执行100000个请求，读的速度是110000次/s，写的速度是81000次/s，且Redis通过提供多种键值数据类型来适应不同场景下的存储需求，目前为止Redis支持的键值数据类型如下：

- 1) 字符串类型 string
- 2) 哈希类型 hash
- 3) 列表类型 list
- 4) 集合类型 set
- 5) 有序集合类型 sortedset

#### 1.3.1 redis的应用场景

- 缓存（数据查询、短连接、新闻内容、商品内容等等）
- 聊天室的在线好友列表
- 任务队列。（秒杀、抢购、12306等等）
- 应用排行榜
- 网站访问统计
- 数据过期处理（可以精确到毫秒）
- 分布式集群架构中的session分离

## 2. 下载安装

1. 官网: <https://redis.io>
2. 中文网: <http://www.redis.net.cn/>
3. 解压直接可以使用：
  - \* redis.windows.conf: 配置文件
  - \* redis-cli.exe: redis的客户端
  - \* redis-server.exe: redis服务器端

## 3. 命令操作

### 1. redis的数据结构：

- \* redis存储的是：key,value格式的数据，其中key都是字符串，value有5种不同的数据结构
- \* value的数据结构：
  - 1) 字符串类型 string
  - 2) 哈希类型 hash : map格式

- 3) 列表类型 list : linkedlist格式。支持重复元素
- 4) 集合类型 set : 不允许重复元素
- 5) 有序集合类型 sortedset: 不允许重复元素, 且元素有顺序

## 2. 字符串类型 string

- 1. 存储: set key value  
127.0.0.1:6379> set username zhangsan  
OK
- 2. 获取: get key  
127.0.0.1:6379> get username  
"zhangsan"
- 3. 删除: del key  
127.0.0.1:6379> del age  
(integer) 1

## 3. 哈希类型 hash

- 1. 存储: hset key field value  
127.0.0.1:6379> hset myhash username lisi  
(integer) 1  
127.0.0.1:6379> hset myhash password 123  
(integer) 1
- 2. 获取:
  - \* hget key field: 获取指定的field对应的值  
127.0.0.1:6379> hget myhash username  
"lisi"
  - \* hgetall key: 获取所有的field和value  
127.0.0.1:6379> hgetall myhash  
1) "username"  
2) "lisi"  
3) "password"  
4) "123"
- 3. 删除: hdel key field  
127.0.0.1:6379> hdel myhash username  
(integer) 1

## 4. 列表类型 list:可以添加一个元素到列表的头部(左边)或者尾部(右边)

- 1. 添加:
  - 1. lpush key value: 将元素加入列表左表
  - 2. rpush key value: 将元素加入列表右边

```
127.0.0.1:6379> lpush myList a
(integer) 1
127.0.0.1:6379> lpush myList b
(integer) 2
127.0.0.1:6379> rpush myList c
(integer) 3
```

- 2. 获取:
  - \* lrange key start end : 范围获取  
127.0.0.1:6379> lrange myList 0 -1  
1) "b"  
2) "a"  
3) "c"

### 3. 删除:

- \* lpop key: 删除列表最左边的元素, 并将元素返回
- \* rpop key: 删除列表最右边的元素, 并将元素返回

### 5. 集合类型 set : 不允许重复元素

#### 1. 存储: sadd key value

```
127.0.0.1:6379> sadd myset a
(integer) 1
127.0.0.1:6379> sadd myset a
(integer) 0
```

#### 2. 获取: smembers key: 获取set集合中所有元素

```
127.0.0.1:6379> smembers myset
1) "a"
```

#### 3. 删除: srem key value: 删除set集合中的某个元素

```
127.0.0.1:6379> srem myset a
(integer) 1
```

6. 有序集合类型 sortedset: 不允许重复元素, 且元素有顺序. 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。

#### 1. 存储: zadd key score value

```
127.0.0.1:6379> zadd mysort 60 zhangsan
(integer) 1
127.0.0.1:6379> zadd mysort 50 lisi
(integer) 1
127.0.0.1:6379> zadd mysort 80 wangwu
(integer) 1
```

#### 2. 获取: zrange key start end [withscores]

```
127.0.0.1:6379> zrange mysort 0 -1
1) "lisi"
2) "zhangsan"
3) "wangwu"

127.0.0.1:6379> zrange mysort 0 -1 withscores
1) "zhangsan"
2) "60"
3) "wangwu"
4) "80"
5) "lisi"
6) "500"
```

#### 3. 删除: zrem key value

```
127.0.0.1:6379> zrem mysort lisi
(integer) 1
```

### 7. 通用命令

1. keys \* : 查询所有的键
2. type key : 获取键对应的value的类型
3. del key: 删除指定的key value

### 4. 持久化

1. redis是一个内存数据库, 当redis服务器重启, 获取电脑重启, 数据会丢失, 我们可以将redis内存中的数据持久化保存到硬盘的文件中。

#### 2. redis持久化机制:

1. RDB: 默认方式, 不需要进行配置, 默认就使用这种机制

\* 在一定的间隔时间中, 检测key的变化情况, 然后持久化数据

1. 编辑redis.windows.conf文件

```
# after 900 sec (15 min) if at least 1 key changed
save 900 1

# after 300 sec (5 min) if at least 10 keys changed
save 300 10

# after 60 sec if at least 10000 keys changed
save 60 10000
```

2. 重新启动redis服务器, 并指定配置文件名称

```
D:\JavaWeb2018\day23_redis\资料\redis\windows-64\redis-2.8.9>redis-server.exe
redis.windows.conf
```

2. AOF: 日志记录的方式, 可以记录每一条命令的操作。可以每一次命令操作后, 持久化数据

1. 编辑redis.windows.conf文件

appendonly no (关闭aof) --> appendonly yes (开启aof)

```
# appendfsync always : 每一次操作都进行持久化
appendfsync everysec : 每隔一秒进行一次持久化
# appendfsync no      : 不进行持久化
```

5. Java客户端 Jedis

\* Jedis: 一款java操作Redis数据库的工具.

\* 使用步骤:

1. 下载jedis的jar包

2. 使用

```
//1. 获取连接
Jedis jedis = new Jedis("localhost",6379);
//2. 操作
jedis.set("username","zhangsan");
//3. 关闭连接
jedis.close();
```

\* Jedis操作各种redis中的数据结构

1) 字符串类型 string

```
set
get
```

//1. 获取连接

```
Jedis jedis = new Jedis();//如果使用空参构造, 默认值 "localhost",6379端口
```

//2. 操作

//存储

```
jedis.set("username","zhangsan");
```

//获取

```
String username = jedis.get("username");
```

```
System.out.println(username);
```

//可以使用setex()方法存储可以指定过期时间的 key value

jedis.setex("activecode",20,"hehe");//将activecode: hehe键值对存入redis, 并且20秒后自动删除该键值对

//3. 关闭连接

```
jedis.close();
```

## 2) 哈希类型 hash : map格式

```
hset
hget
hgetAll
//1. 获取连接
Jedis jedis = new Jedis();//如果使用空参构造, 默认值 "localhost",6379端口
//2. 操作
// 存储hash
jedis.hset("user","name","lisi");
jedis.hset("user","age","23");
jedis.hset("user","gender","female");

// 获取hash
String name = jedis.hget("user", "name");
System.out.println(name);
```

```
// 获取hash的所有map中的数据
Map<String, String> user = jedis.hgetAll("user");

// keyset
Set<String> keySet = user.keySet();
for (String key : keySet) {
    //获取value
    String value = user.get(key);
    System.out.println(key + ":" + value);
}

//3. 关闭连接
jedis.close();
```

## 3) 列表类型 list : linkedlist格式。支持重复元素

```
lpush / rpush
lpop / rpop
lrange start end : 范围获取

//1. 获取连接
Jedis jedis = new Jedis();//如果使用空参构造, 默认值 "localhost",6379端口
//2. 操作
// list 存储
jedis.lpush("mylist","a","b","c");//从左边存
jedis.rpush("mylist","a","b","c");//从右边存

// list 范围获取
List<String> mylist = jedis.lrange("mylist", 0, -1);
System.out.println(mylist);

// list 弹出
String element1 = jedis.lpop("mylist");//c
System.out.println(element1);
```

```
String element2 = jedis.rpop("mylist");//c
System.out.println(element2);

// list 范围获取
List<String> mylist2 = jedis.lrange("mylist", 0, -1);
System.out.println(mylist2);

//3. 关闭连接
jedis.close();
```

#### 4) 集合类型 set ： 不允许重复元素

sadd  
smembers:获取所有元素

```
//1. 获取连接
Jedis jedis = new Jedis();//如果使用空参构造, 默认值 "localhost",6379端口
//2. 操作
```

```
// set 存储
jedis.sadd("myset","java","php","c++");

// set 获取
Set<String> myset = jedis.smembers("myset");
System.out.println(myset);
```

```
//3. 关闭连接
jedis.close();
```

#### 5) 有序集合类型 sortedset: 不允许重复元素, 且元素有顺序

zadd  
zrange

```
//1. 获取连接
Jedis jedis = new Jedis();//如果使用空参构造, 默认值 "localhost",6379端口
//2. 操作
```

```
// sortedset 存储
jedis.zadd("mysortedset",3,"亚瑟");
jedis.zadd("mysortedset",30,"后裔");
jedis.zadd("mysortedset",55,"孙悟空");
```

```
// sortedset 获取
Set<String> mysortedset = jedis.zrange("mysortedset", 0, -1);

System.out.println(mysortedset);
```

```
//3. 关闭连接
jedis.close();
```

\* jedis连接池: JedisPool

\* 使用:

1. 创建JedisPool连接池对象
2. 调用方法 getResource()方法获取Jedis连接

```

//0.创建一个配置对象
JedisPoolConfig config = new JedisPoolConfig();
config.setMaxTotal(50);
config.setMaxIdle(10);

//1.创建Jedis连接池对象
JedisPool jedisPool = new JedisPool(config,"localhost",6379);

//2.获取连接
Jedis jedis = jedisPool.getResource();
//3. 使用
jedis.set("hehe","heihei");

```

```

//4. 关闭 归还到连接池中
jedis.close();

```

\* 连接池工具类

```

public class JedisPoolUtils {

    private static JedisPool jedisPool;

    static{
        //读取配置文件
        InputStream is =
JedisPoolUtils.class.getClassLoader().getResourceAsStream("jedis.properties");
        //创建Properties对象
        Properties pro = new Properties();
        //关联文件
        try {
            pro.load(is);
        } catch (IOException e) {
            e.printStackTrace();
        }
        //获取数据, 设置到JedisPoolConfig中
        JedisPoolConfig config = new JedisPoolConfig();
        config.setMaxTotal(Integer.parseInt(pro.getProperty("maxTotal")));
        config.setMaxIdle(Integer.parseInt(pro.getProperty("maxIdle")));

        //初始化JedisPool
        jedisPool = new
JedisPool(config,pro.getProperty("host"),Integer.parseInt(pro.getProperty("port")));

```

```

    }

```

```

/**
 * 获取连接方法
 */
public static Jedis getJedis(){
    return jedisPool.getResource();
}

}

```



## 案例：

案例需求：

1. 提供index.html页面，页面中有一个省份 下拉列表
2. 当 页面加载完成后 发送ajax请求，加载所有省份

\* 注意：使用redis缓存一些不经常发生变化的数据。

\* 数据库的数据一旦发生改变，则需要更新缓存。

\* 数据库的表执行 增删改的相关操作，需要将redis缓存数据情况，再次存入

\* 在service对应的增删改方法中，将redis数据删除。