1. Redis
2. 概述

Redis 高性能的NOSQL系列的菲关系型数据库。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。

1. Redis支持的数据类型
2. 字符串类型 String
3. 哈希类型 hash
4. 列表类型 list
5. 集合类型 set
6. 有序集合类型 sortedset
7. Redis的应用场景
8. 缓存（数据查询，短连接，内容）
9. 任务队列（秒杀，抢购）
10. 应用排行榜
11. 数据过期处理
12. 分布式集群架构中的session分离
13. 安装与使用
14. Redis下载

大陆地区请到Redis中文网<http://www.redis.net.cn/>，下载redis。

1. 使用

解压下载的压缩包，可以得到一些文件：

redis.windows.conf是配置文件

redis-cli.ext是redis的客户端

redis-server.exe是redis的服务器端

要使用redis，，先要打开服务器端，再打开客户端连接到服务器。

1. Redis的数据结构

Redis存储的是键值对的数据，key都是字符串，value有5种不同类型：

字符串类型 String

哈希类型 hash

列表类型 list

集合类型 set

有序集合类型 sortedset

1. Redis的命令操作
2. 字符串类型String
3. 存储

set key value

如：set age 22

1. 获取

get key

如：

get age

1. 删除

del key

如：del age

1. 哈希类型 hash
2. 存储

hset key field value

如：field value是一个键值对

hset user name 22

1. 获取

hget key field

获取指定的field对应的值

如：hget user name

hgetall key

获取所有的field和value

如：hgetall user

1. 删除

hdel key field

如：hdel user name

1. 列表类型list
2. 添加

可以添加一个元素到列表的头部（左边）或者尾部（右边）

lpush key value 将元素添加到列表的左边

rpush key value 将元素添加到列表的右边

如：lpush mylist 1

rpush mylist 10

1. 获取

可以通过一个范围获取元素：

lrange key start end

如：

lrange mylist 1 2

1. 删除

lpop key 删除列表最左边的元素，并返回该元素

rpop key 删除列表最右边的元素，并返回该元素

1. 集合类型set

set集合不允许有重复的元素。

1. 存储

sadd key value

如：

sadd age 22

1. 获取

smembers key 获取set集合中的所有元素

如：

smembers age

1. 删除

srem key value， 删除set集合中的某个元素

srem age 22

1. 有序集合类型sortedset

不仅不允许重复元素，且元素有序

1. 存储

zadd key score value， 通过score分数来排序

如：

zadd age 80 22

1. 获取

zrange key start end

如：

zrange age 0 -1

1. 删除

zrem key value

如：

zrem age 22

1. 通用命令
2. keys \*

查询所有的键

1. type key

获取键对应的value的类型

1. del key

删除指定的key value

1. 持久化
2. 概述

Redis是一个内存数据库，当Redis服务器重启或关闭时，数据就会丢失，我们需要将Redis内存中的数据持久化到硬盘上。

1. Redis持久化机制
2. RDB

Redis持久化的默方式，不需要进行配置，在一定的间隔时间中，会自动检测key的变化情况，然后持久化数据。

想要修改RDB的间隔时间，可以到conf配置文件中进行修改

找到如下的字段：

save 900 1 ：至少一个key变化时900s后持久化

save 300 10 ：至少10个key变化时300s后持久化

save 60 10000 ：至少10000个key变化时，60s后持久化

通过注释来阅读并修改。修改后，重启服务器，并制定配置文件名称。

cd 到redis的文件夹，执行redis-server.exe命令并 添加 配置文件，如：

redis-server.exe redis.windows.conf

1. AOF

通过日志记录的方式，可以记录每一条命令的操作，在执行每一条Redis命令后，都会持久化数据。

同样的，我们可以在conf配置文件中修改AOF的配置：

appendonly no 表示着关闭AOF

appendonly yes 表示着开启AOF

appendfsync always 表示着每一次操作有持久化

appendfsync everysec 表示这每隔一秒进行一次持久化

appendfsync no 表示这不进行持久化

1. NOSQL

（一）概述

Not Only SQL，泛指非关系型数据库。随着Web2.0的兴起，传统的关系型数据库对于超大规模与高并发的Web2.0网站已经显得力不从心了。菲关系型数据库可以用作缓存来解决大数据的应用难题。由于NOSQL将数据存储在缓存中，查询速度会非常快。且NOSQL的存储格式有很多种，如key ,value,文档，图片。

（二）NOSQL与关系型数据库的优劣

1. 菲关系型数据库的优势
2. 性能NOSQL是基于键值对的，可以想象成表中的主键和之的对应关系，不需要经过SQL层的解析，性能很高。
3. 可扩展性也是基于键值对，数据之间没有耦合性，所有容易扩展。
4. 关系型数据库的优势
5. 复杂查询可以用SQL语句方便地在一个表以及多个表之间做复杂查询。
6. 事务支持使得对于安全性能很高的数据访问得以实现。
7. Jedis
8. 概述

如同我们之前使用JDBC连接MySql数据库一样，Jedis用于Java操作Redis数据库。

1. 使用步骤
2. 导入jedis的jar包
3. 通过Java编写程序操作Redis数据库

package redis;  
  
import org.junit.Test;  
import redis.clients.jedis.Jedis;  
  
*/\*\*  
 \* jedis的测试类  
 \*/*public class RedisTest {  
  
 @Test  
 public void test1(){  
 //获取连接  
 Jedis jedis = new Jedis("localhost", 6379);  
 //操作  
 jedis.set("username", "hehe");  
 //关闭连接  
 jedis.close();  
  
 }  
  
}

1. 字符串的操作

Jedis中的方法名与Redis的操作是相同的。

set存储,get获取,setex存储有时间限制的键值对。

Jedis jedis = new Jedis();

jedis.set(“username”, “hehe”);

String username = jedis.get(“username”);

jedis.setex(“activecode”, 20, “hehe”); //20秒后删除

jedis.close();

1. 哈希类型的操作

hset, hget。

jedis.hset(“user”, “name”, “eee”);

jedis.hset(“user”, “age”, “23”);

String name – jedis.hget(“user”, “name”);

//获取hash中的所有map中的数据

Map<String ,String> user = jedis.hgetAll(“user”);

Set<String> keyset = user.keySet();

for(String key : keySet){

String value = user.get(key);

System.out.println(key);

}

1. 列表list的操作

lpush, rpush, lpop, rpop

jedis.lpush(“mylist”, “a”);

jedis.rpush(“mylist”, “b”);

List<String> mylist = jedis.lrange(“mylist”, 0, -1);

1. 集合set的操作

sadd，smembers

jedis.sadd(“myset”, “java”);

jedis.sass(“myset”, “c”);

Set<String> myset = jedis.smembers(“myset”);

1. Jedis连接池

Jedis内置了连接池JedisPool，使用方法非常简单：

首先创建JedisPool连接池对象，再调用getResource()方法获取Jedis连接。

JedisPool jedisPool = new JedisPool();

Jedis jedis = jedisPool.getResource();

jedis.set(“name”, “hehe”);

jedis.close();

可以通过配置对象来配置JedisPool。在创建JedsiPool连接池对象时要将该配置对象传进去。

JedisPoolConfig config = new JedisPoolConfig();

config.setMaxTotal(50);

config.setMaxIdle(10);

JedisPool jedisPool = new JedisPool(config, “localhost”, 6379);

也可以通过设置一个RedisPoolUtils工具类专用于连接Redis数据库以及通过redis.properties配置文件来配置RedisPool连接池。

package utils;  
  
import redis.clients.jedis.Jedis;  
import redis.clients.jedis.JedisPool;  
import redis.clients.jedis.JedisPoolConfig;  
import redis.clients.util.JedisURIHelper;  
  
import java.io.IOException;  
import java.io.InputStream;  
import java.util.Properties;  
  
public class JetisUtils {  
  
 private static JedisPool *jedisPool*;  
 static {  
 //读取配置文件  
 InputStream is = JetisUtils.class.getClassLoader().getResourceAsStream("jedis.properties");  
 //创建Properties对象  
 Properties pro = new Properties();  
 //冠梁文件  
 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")));  
  
  
 }  
  
 public static Jedis getJedis(){  
 return *jedisPool*.getResource();  
 }  
}