Redis教程

1. **概述**

REmote DIctionary Server(Redis) 是一个由Salvatore Sanfilippo写的key-value存储系统。

Redis是一个开源的使用ANSI C语言编写、遵守BSD协议、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。

它通常被称为数据结构服务器，因为值（value）可以是 字符串(String), 哈希(Map), 列表(list), 集合(sets) 和 有序集合(sorted sets)等类型。

特点：

* Redis支持数据的持久化，可以将内存中的数据保存在磁盘中，重启的时候可以再次加载进行使用。
* Redis不仅仅支持简单的key-value类型的数据，同时还提供list，set，zset，hash等数据结构的存储。
* Redis支持数据的备份，即master-slave模式的数据备份。

优势：

 性能极高 – Redis能读的速度是110000次/s,写的速度是81000次/s 。

 丰富的数据类型 – Redis支持二进制案例的 Strings, Lists, Hashes, Sets 及 Ordered Sets 数据类型操作。

 原子 – Redis的所有操作都是原子性的，意思就是要么成功执行要么失败完全不执行。单个操作是原子性的。多个操作也支持事务，即原子性，通过MULTI和EXEC指令包起来。

 丰富的特性 – Redis还支持 publish/subscribe, 通知, key 过期等等特性

优点：

1 读写性能优异

2 支持数据持久化，支持AOF和RDB两种持久化方式

3 支持主从复制，主机会自动将数据同步到从机，可以进行读写分离。

4 数据结构丰富：除了支持string类型的value外还支持string、hash、set、sortedset、list等数据结构。

缺点：

1 Redis不具备自动容错和恢复功能，主机从机的宕机都会导致前端部分读写请求失败，需要等待机器重启或者手动切换前端的IP才能恢复。

2 主机宕机，宕机前有部分数据未能及时同步到从机，切换IP后还会引入数据不一致的问题，降低了系统的可用性。

3 Redis的主从复制采用全量复制，复制过程中主机会fork出一个子进程对内存做一份快照，并将子进程的内存快照保存为文件发送给从机，这一过程需要确保主机有足够多的空余内存。若快照文件较大，对集群的服务能力会产生较大的影响，而且复制过程是在从机新加入集群或者从机和主机网络断开重连时都会进行，也就是网络波动都会造成主机和从机间的一次全量的数据复制，这对实际的系统运营造成了不小的麻烦。

4 Redis较难支持在线扩容，在集群容量达到上限时在线扩容会变得很复杂。为避免这一问题，运维人员在系统上线时必须确保有足够的空间，这对资源造成了很大的浪费。

1. **什么是BSD协议**

BSD开源协议是一个给于使用者很大自由的协议。可以自由的使用，修改源代码，也可以将修改后的代码作为开源或者专有软件再发布。当你发布使用了BSD协议的代码，或者以BSD协议代码为基础做二次开发自己的产品时，需要满足三个条件：

* 如果再发布的产品中包含源代码，则在源代码中必须带有原来代码中的BSD协议。
* 如果再发布的只是二进制类库/软件，则需要在类库/软件的文档和版权声明中包含原来代码中的BSD协议。
* 不可以用开源代码的作者/机构名字和原来产品的名字做市场推广。

BSD代码鼓励代码共享，但需要尊重代码作者的著作权。BSD由于允许使用者修改和重新发布代码，也允许使用或在BSD代码上开发商业软件发布和销 售，因此是对商业集成很友好的协议。

很多的公司企业在选用开源产品的时候都首选BSD协议，因为可以完全控制这些第三方的代码，在必要的时候可以修改或者 二次开发。

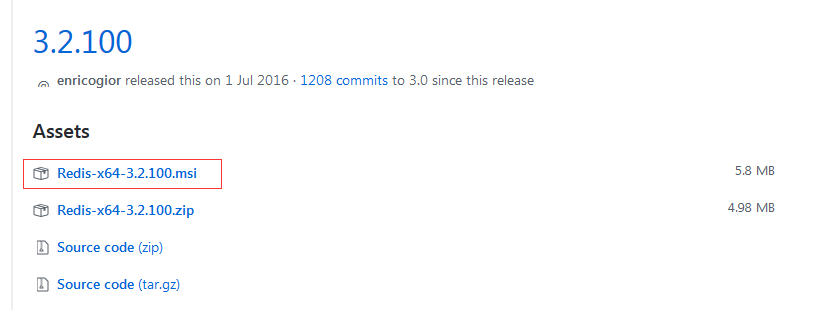
1. **下载安装**

**下载地址：**官网上可以下载的windows版本的，现在官网以及没有下载地址，只能在github上下载，官网只提供linux版本的下载

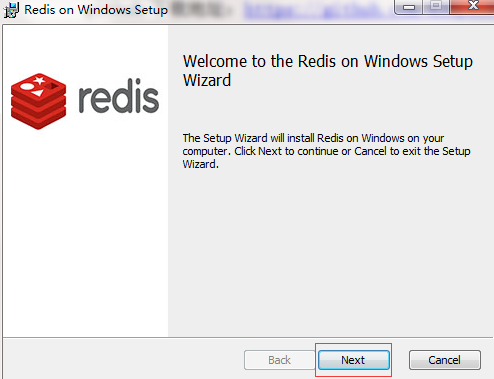
官网下载地址：http://redis.io/download

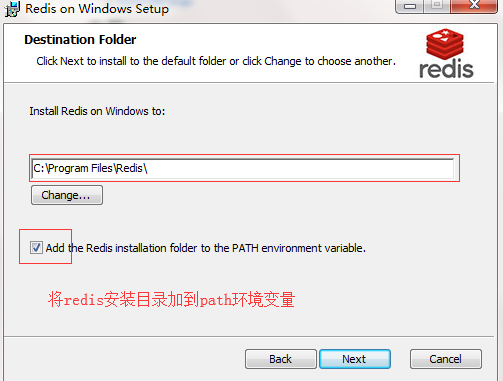
github下载地址：<https://github.com/MSOpenTech/redis/tags>

安装版本如下：

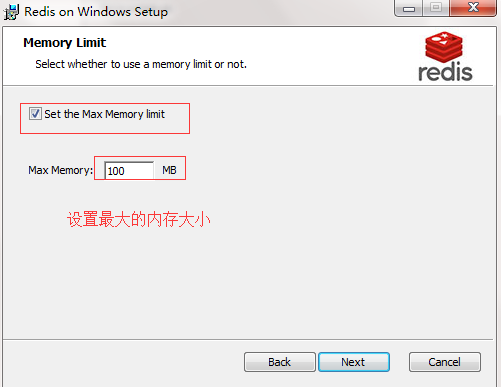


**安装：**

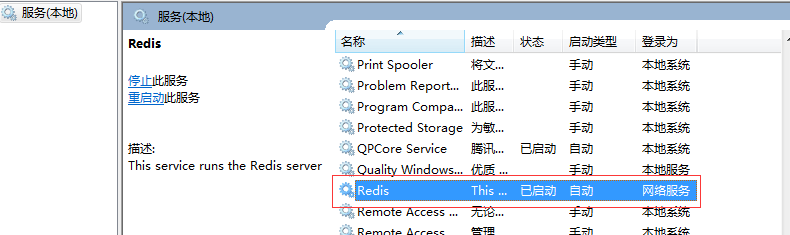








安装成功后，可以查看Windows服务里面能看到redis这时已经启动服务



测试安装成功后的redis是否运行正常

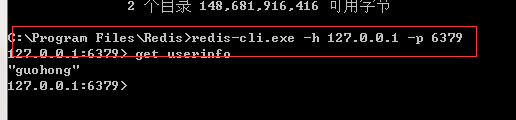
打开dos命令，进入到redis的安装目录，然后输入如下命令

redis-cli.exe –h 127.0.0.1 –p 6379

如果能连接上redis说明安装成功

set userinfo guohong 设置key为userinfo键值为guohong的key-val

get userinfo 查看键为userinfo的值

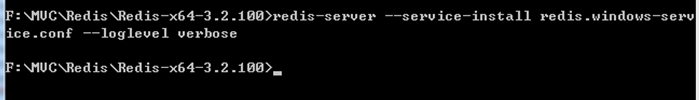


通过Windows命令启动服务，该命令在redis的安装目录下执行

redis-server redis.windows.conf

如果我们安装后，在Windows的服务里面没有redis的服务，则需要我们手动加入到服务中

redis-server --service-install redis.windows-service.conf --loglevel verbose



常用服务命令：

卸载服务：redis-server --service-uninstall

开启服务：redis-server --service-start

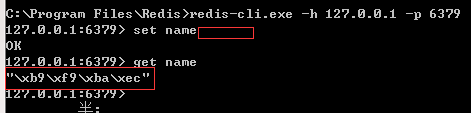
停止服务：redis-server --service-stop

1. **查看redis的值为中文的情况**

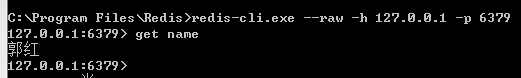
使用如下命令，连接redis的时候加上—raw

redis-cli.exe --raw -h 127.0.0.1 -p 6379

没加—raw的效果如下，设置name的值为郭红，这里可能是dos的问题，中文值没显示出来，在get的时候拿到的就不是中文



加上后的效果



1. **安全设置-设置密码连接**

以下都是用的命令操作

步骤1：连接redis

redis-cli.exe -h 127.0.0.1 -p 6379

步骤2：

CONFIG GET requirepass 查看是否设置了密码



步骤3：设置密码，密码为guohong

CONFIG SET requirepass "guohong"



这个时候再进行操作会提示你没有权限，需要验证密码



步骤4：验证密码

AUTH guohong



然后再查看密码，此时已经有了密码



1. **Redis的demo**

我的是maven工程

所以在pom.xml文件中还需要配置如下：

<dependencies>

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

<version>2.9.0</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.9</version>

<scope>test</scope>

</dependency>

</dependencies>

Redis连接类：

package com.gh;

import redis.clients.jedis.Jedis;

import redis.clients.jedis.JedisPool;

import redis.clients.jedis.JedisPoolConfig;

public final class RedisPool {

// Redis服务器IP

private static String ADDR = "127.0.0.1";

// Redis的端口号

private static Integer PORT = 6379;

// 访问密码

private static String AUTH = "guohong";

// 可用连接实例的最大数目，默认为8；

// 如果赋值为-1，则表示不限制，如果pool已经分配了maxActive个jedis实例，则此时pool的状态为exhausted(耗尽)

private static Integer MAX\_TOTAL = 1024;

// 控制一个pool最多有多少个状态为idle(空闲)的jedis实例，默认值是8

private static Integer MAX\_IDLE = 200;

// 等待可用连接的最大时间，单位是毫秒，默认值为-1，表示永不超时。

// 如果超过等待时间，则直接抛出JedisConnectionException

private static Integer MAX\_WAIT\_MILLIS = 10000;

private static Integer TIMEOUT = 10000;

// 在borrow(用)一个jedis实例时，是否提前进行validate(验证)操作；

// 如果为true，则得到的jedis实例均是可用的

private static Boolean TEST\_ON\_BORROW = true;

private static JedisPool jedisPool = null;

/\*\*

\* 静态块，初始化Redis连接池

\*/

static {

try {

JedisPoolConfig config = new JedisPoolConfig();

/\*

\* 注意： 在高版本的jedis

\* jar包，比如本版本2.9.0，JedisPoolConfig没有setMaxActive和setMaxWait属性了

\* 这是因为高版本中官方废弃了此方法，用以下两个属性替换。 maxActive ==> maxTotal maxWait==>

\* maxWaitMillis

\*/

config.setMaxTotal(MAX\_TOTAL);

config.setMaxIdle(MAX\_IDLE);

config.setMaxWaitMillis(MAX\_WAIT\_MILLIS);

config.setTestOnBorrow(TEST\_ON\_BORROW);

jedisPool = new JedisPool(config,ADDR,PORT,TIMEOUT,AUTH);

//无密码的连接

//jedisPool = new JedisPool(config, ADDR, PORT, TIMEOUT);

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* 获取Jedis实例

\*

\* @return

\*/

public synchronized static Jedis getJedis() {

try {

if (jedisPool != null) {

Jedis jedis = jedisPool.getResource();

return jedis;

} else {

return null;

}

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

public static void returnResource(final Jedis jedis) {

// 方法参数被声明为final，表示它是只读的。

if (jedis != null) {

jedisPool.returnResource(jedis);

// jedis.close()取代jedisPool.returnResource(jedis)方法将3.0版本开始

// jedis.close();

}

}

}

测试用例：

package com.gh;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import org.junit.Before;

import org.junit.Test;

import redis.clients.jedis.Jedis;

public class TestRedis {

private Jedis jedis;

@Before

public void setJedis() {

//连接redis服务器(在这里是连接本地的)

jedis = new Jedis("127.0.0.1", 6379);

//权限认证

jedis.auth("guohong");

System.out.println("连接服务成功");

}

/\*\*

\* Redis操作字符串

\*/

@Test

public void testString() {

//添加数据

jedis.set("name", "chx"); //key为name放入value值为chx

System.out.println("拼接前:" + jedis.get("name"));//读取key为name的值

//向key为name的值后面加上数据 ---拼接

jedis.append("name", " is my name;");

System.out.println("拼接后:" + jedis.get("name"));

//删除某个键值对

jedis.del("name");

System.out.println("删除后:" + jedis.get("name"));

//s设置多个键值对

jedis.mset("name", "chenhaoxiang", "age", "20", "email", "chxpostbox@outlook.com");

jedis.incr("age");//用于将键的整数值递增1。如果键不存在，则在执行操作之前将其设置为0。 如果键包含错误类型的值或包含无法表示为整数的字符串，则会返回错误。此操作限于64位有符号整数。

System.out.println(jedis.get("name") + " " + jedis.get("age") + " " + jedis.get("email"));

}

@Test

public void testMap() {

//添加数据

Map<String, String> map = new HashMap<String, String>();

map.put("name", "chx");

map.put("age", "100");

map.put("email", "\*\*\*@outlook.com");

jedis.hmset("user", map);

//取出user中的name，结果是一个泛型的List

//第一个参数是存入redis中map对象的key，后面跟的是放入map中的对象的key，后面的key是可变参数

List<String> list = jedis.hmget("user", "name", "age", "email");

System.out.println(list);

//删除map中的某个键值

jedis.hdel("user", "age");

System.out.println("age:" + jedis.hmget("user", "age")); //因为删除了，所以返回的是null

System.out.println("user的键中存放的值的个数:" + jedis.hlen("user")); //返回key为user的键中存放的值的个数2

System.out.println("是否存在key为user的记录:" + jedis.exists("user"));//是否存在key为user的记录 返回true

System.out.println("user对象中的所有key:" + jedis.hkeys("user"));//返回user对象中的所有key

System.out.println("user对象中的所有value:" + jedis.hvals("user"));//返回map对象中的所有value

//拿到key，再通过迭代器得到值

Iterator<String> iterator = jedis.hkeys("user").iterator();

while (iterator.hasNext()) {

String key = iterator.next();

System.out.println(key + ":" + jedis.hmget("user", key));

}

jedis.del("user");

System.out.println("删除后是否存在key为user的记录:" + jedis.exists("user"));//是否存在key为user的记录

}

/\*\*

\* jedis操作List

\*/

@Test

public void testList(){

//移除javaFramwork所所有内容

jedis.del("javaFramwork");

//存放数据

jedis.lpush("javaFramework","spring");

jedis.lpush("javaFramework","springMVC");

jedis.lpush("javaFramework","mybatis");

//取出所有数据,jedis.lrange是按范围取出

//第一个是key，第二个是起始位置，第三个是结束位置

System.out.println("长度:"+jedis.llen("javaFramework"));

//jedis.llen获取长度，-1表示取得所有

System.out.println("javaFramework:"+jedis.lrange("javaFramework",0,-1));

jedis.del("javaFramework");

System.out.println("删除后长度:"+jedis.llen("javaFramework"));

System.out.println(jedis.lrange("javaFramework",0,-1));

}

/\*\*

\* jedis操作Set

\*/

@Test

public void testSet(){

//添加

jedis.sadd("user","chenhaoxiang");

jedis.sadd("user","hu");

jedis.sadd("user","chen");

jedis.sadd("user","xiyu");

jedis.sadd("user","chx");

jedis.sadd("user","are");

//移除user集合中的元素are

jedis.srem("user","are");

System.out.println("user中的value:"+jedis.smembers("user"));//获取所有加入user的value

System.out.println("chx是否是user中的元素:"+jedis.sismember("user","chx"));//判断chx是否是user集合中的元素

System.out.println("集合中的一个随机元素:"+jedis.srandmember("user"));//返回集合中的一个随机元素

System.out.println("user中元素的个数:"+jedis.scard("user"));

}

/\*\*

\* 排序

\*/

@Test

public void test(){

jedis.del("number");//先删除数据，再进行测试

jedis.rpush("number","4");//将一个或多个值插入到列表的尾部(最右边)

jedis.rpush("number","5");

jedis.rpush("number","3");

jedis.lpush("number","9");//将一个或多个值插入到列表头部

jedis.lpush("number","1");

jedis.lpush("number","2");

System.out.println(jedis.lrange("number",0,jedis.llen("number")));

System.out.println("排序:"+jedis.sort("number"));

System.out.println(jedis.lrange("number",0,-1));//不改变原来的排序

jedis.del("number");//测试完删除数据

}

}

1. **a**