

day39-redis

学习目标：

1. 能够理解nosql的概念
2. 能够说出redis的常用数据类型
3. 能够使用redis的string操作命令
4. 能够使用redis的hash操作命令
5. 能够使用redis的list操作命令
6. 能够使用redis的set操作命令
7. 能够说出redis的两种持久化机制
8. 能够使用jedis对redis进行操作

一, nosql概述

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。

MySql: 关系型的数据库

Oracle: 关系型的数据库

2.为什么需要NOSQL

随着互联网的高速崛起, 网站的用户群的增加, 访问量的上升, 传统数据库上都开始出现了性能瓶颈, web程序不再仅仅专注在功能上, 同时也在追求性能。所以NOSQL数据库应运而生, 具体表现为对如下三高问题的解决:

- High performance - 对数据库高并发读写的需求

web2.0网站要根据用户个性化信息来实时生成动态页面和提供动态信息, 所以基本上无法使用动态页面静态化技术, 因此数据库并发负载非常高, 往往要达到每秒上万次读写请求。关系数据库应付上万次SQL查询还勉强顶得住, 但是应付上万次SQL写数据请求, 硬盘IO就已经无法承受了。其实对于普通的BBS网站, 往往也存在对高并发写请求的需求, 例如网站的实时统计在线用户状态, 记录热门帖子的点击次数, 投票计数等, 因此这是一个相当普遍的需求。

- Huge Storage - 对海量数据的高效率存储和访问的需求

类似Facebook, twitter, Friendfeed这样的SNS网站, 每天用户产生海量的用户动态, 以Friendfeed为例, 一个月就达到了2.5亿条用户动态, 对于关系数据库来说, 在一张2.5亿条记录的表里面进行SQL查询, 效率是极其低下乃至不可忍受的。再例如大型web网站的用户登录系统, 例如腾讯, 盛大, 动辄数以亿计的帐号, 关系数据库也很难应付。

- High Scalability & High Availability- 对数据库的高可扩展性和高可用性的需求

在基于web的架构当中, 数据库是最难进行横向扩展的, 当一个应用系统的用户量和访问量与日俱增的时候, 你的数据库却没有办法像web server和app server那样简单的通过添加更多的硬件和服务节点来扩展性能和负载能力。对于很多需要提供24小时不间断服务的网站来说, 对数据库系统进行升级和扩展是非常痛苦的事情, 往往需要停机维护和数据迁移, 为什么数据库不能通过不断的添加服务器节点来实现扩展呢?

解决三高: 高并发,大数据查数据, 高可靠和高扩展

3 主流的NOSQL产品

NOSQL 非关系型数据库 ----- 关系型数据库

Redis ----- MySql



- 键值(Key-Value)存储数据库 Redis
- 列存储数据库(分布式)
- 文档型数据库 (Web应用与Key-Value类似，Value是结构化的)
- 图形(Graph)数据库(图结构)

4 NOSQL的特点

在大数据存取上具备关系型数据库无法比拟的性能优势，例如：

- 易扩展
NoSQL数据库种类繁多，但是一个共同的特点都是去掉关系数据库的关系型特性。数据之间无关系，这样就非常容易扩展。也无形之间，在架构的层面上带来了可扩展的能力。
- 大数据量，高性能
NoSQL数据库都具有非常高的读写性能，尤其在大数据量下，同样表现优秀。这得益于它的无关系性，数据库的结构简单。
- 灵活的数据模型
NoSQL无需事先为要存储的数据建立字段，随时可以存储自定义的数据格式。而在关系数据库里，增删字段是一件非常麻烦的事情。如果是非常大数据量的表，增加字段简直就是一个噩梦。这点在大数据量的Web2.0时代尤其明显。
- 高可用
NoSQL在不太影响性能的情况，就可以方便的实现高可用的架构。比如Cassandra，HBase模型，通过复制模型也能实现高可用。

二, Redis概述

1.什么是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：

- 字符串类型 string
- 散列类型 hash
- 列表类型 list
- 集合类型 set
- 有序集合类型 sortedset

2 redis的应用场景

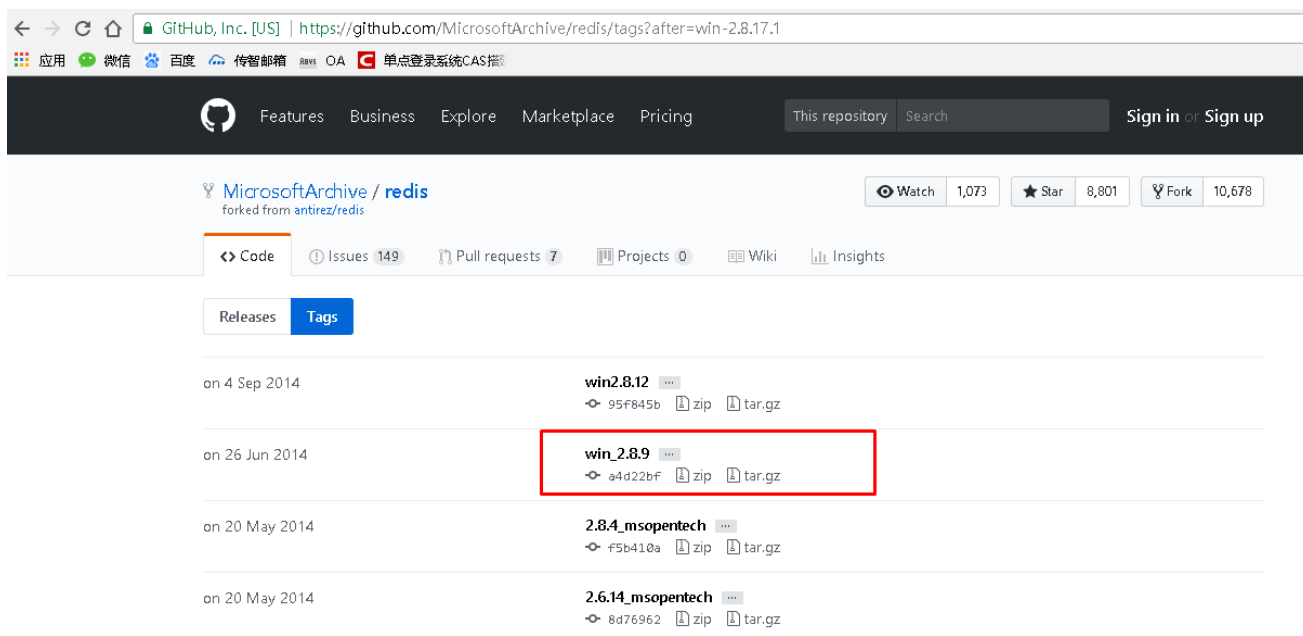
- 缓存（数据查询、短连接、新闻内容、商品内容等等）
- 任务队列。（秒杀、抢购、12306等等）
- 数据过期处理（可以精确到毫秒）
- 聊天室的在线好友列表
- 应用排行榜
- 网站访问统计
- 分布式集群架构中的session分离

三, window版Redis的安装与使用

1.windows版Redis的下载

官方提倡使用Linux版的Redis，所以官网提供了Linux版的Redis下载，我们可以从GitHub上下载window版的Redis，具体链接地址如下：

- 官网下载地址：<http://redis.io/download>
- github下载地址：<https://github.com/MSOpenTech/redis/tags>



在今天的课程资料中提供的下载完毕的window版本的Redis：



redis-2.8.9.zip

2 window版Redis的目录结构

解压Redis压缩包后，见到如下目录机构：

目录或文件	作用
redis-benchmark	性能测试工具
redis-check-aof	AOF文件修复工具
redis-check-dump	RDB文件检查工具（快照持久化文件）
redis-cli	命令行客户端
redis-server	redis服务器启动命令
redis.windows.conf	redis核心配置文件

3 window版Redis的安装与启动

	redis-server.exe	2013/6/10 11:58	应用程序	1	702 KB
	redis-cli.exe	2013/6/10 11:58	应用程序	2	163 KB
	redis-check-dump.exe	2013/6/10 11:58	应用程序		101 KB
	redis-check-aof.exe	2013/6/10 11:58	应用程序		94 KB
	redis-benchmark.exe	2013/6/10 11:58	应用程序		123 KB


- 安装:window版的安装及其简单，解压Redis压缩包完成即安装完毕
- 启动与关闭: 双击Redis目录中redis-server.exe可以启动redis服务，Redis服务占用的端口是6379)



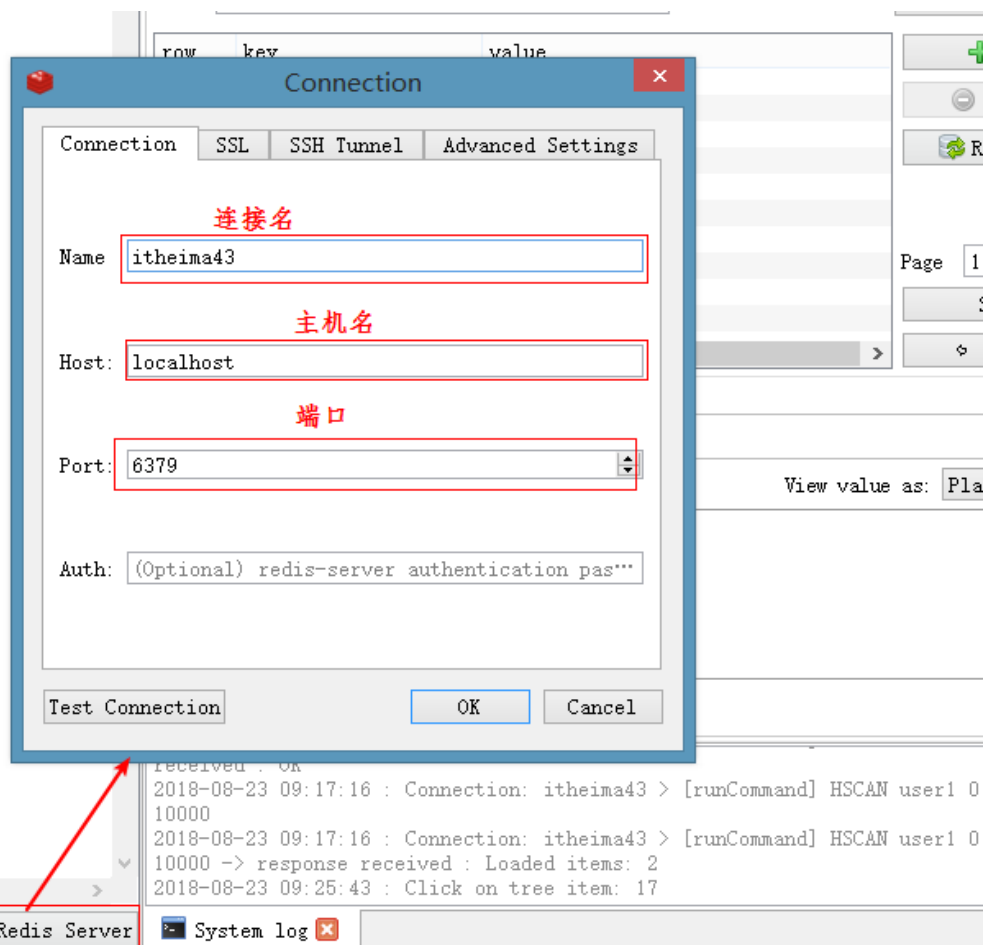
- 点击redis-cli

```
127.0.0.1:6379> set name "itast"  
OK  
127.0.0.1:6379> get name  
"itast"  
127.0.0.1:6379> _
```

4.可视化软件的安装和使用

- 点击安装  [redis-desktop-manager-0.8.0.3841.exe](#)
- 连接

ma16
ma17
ma18
68.147.140
ma19
ma20
ma21
ma22
ma24
ma28
m31
ma35
ma38
工程
ma43



四 Redis的数据类型

1,redis中数据结构

- redis中存储的数据是以key-value的形式存在的,其中value支持5种数据类型.在日常开发中主要使用比较多的有字符串、哈希、字符串列表、字符串集合四种类型,其中最为常用的是字符串类型。

字符串(String) 类似java里面的字符串

哈希(hash) 类似java里面的map

字符串列表(list) 类似java里面的List

字符串集合(set) 类似java里面的Set

有序的字符串集合(sorted-set或者叫zset)

- key不要太长(不能>1024个字节),
- 也不要太短. 可读性差.
- key在项目里面最好统一写法, key的常用的写法:

项目名子模块key名称; eg: travel_user_username

2,存储字符串

2.1概述

string是redis最基本的类型,用的也是最多的，一个key对应一个value。一个键最大能存储512MB。

2.2常见命令

命令	描述
set key value	设置指定 key 的值
get key	获取指定 key 的值
del key	删除key

3.,存储hash

3.1.概述

Redis中hash 是一个**键值对集合**。

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。

Redis存储hash可以看成是String key 和String value的map容器. 也就是说把值看成map集合.

- Eg:

key	value	
user1	name	zs
	age	18
user2	name	ls
	age	19

3.2.常见命令

命令	命令描述
hset key filed value	将哈希表 key 中的字段 field 的值设为 value
hmset key field1 value1 [field2 value2]...	同时将多个 field-value (字段-值)对设置到哈希表 key 中
hget key filed	获取存储在哈希表中指定字段的值
hmget key filed1 [filed2]	获取多个给定字段的值
hdel key filed1 [filed2]	删除一个或多个哈希表字段
hlen key	获取哈希表中字段的数量
del key	删除整个hash(对象)

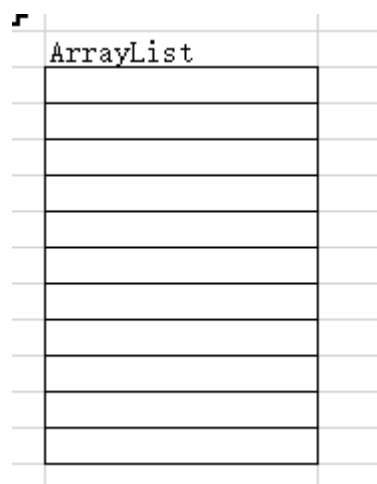
4.存储list

4.1.概述

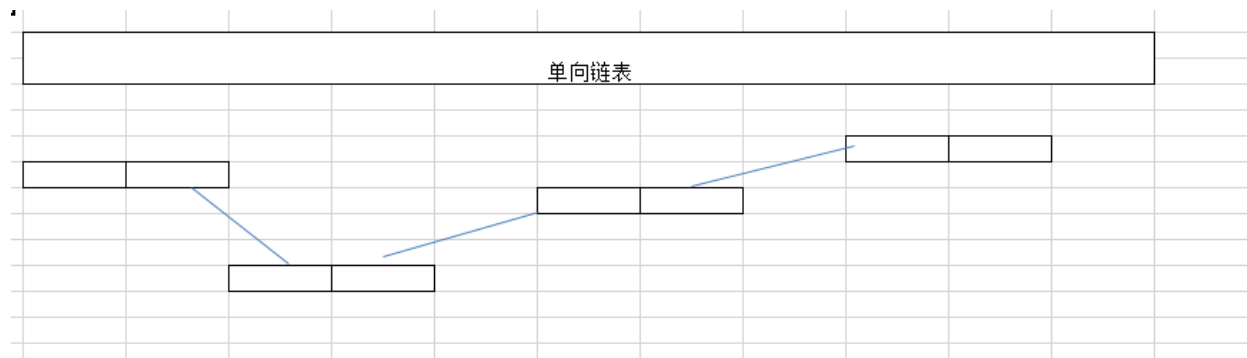
Redis列表是简单的字符串列表，按照**插入顺序排序**。你可以添加一个元素到列表的头部（左边）或者尾部（右边）

一个列表最多可以包含 $2^{32} - 1$ 个元素 (4294967295, 每个列表超过40亿个元素)。 **特点:有序**

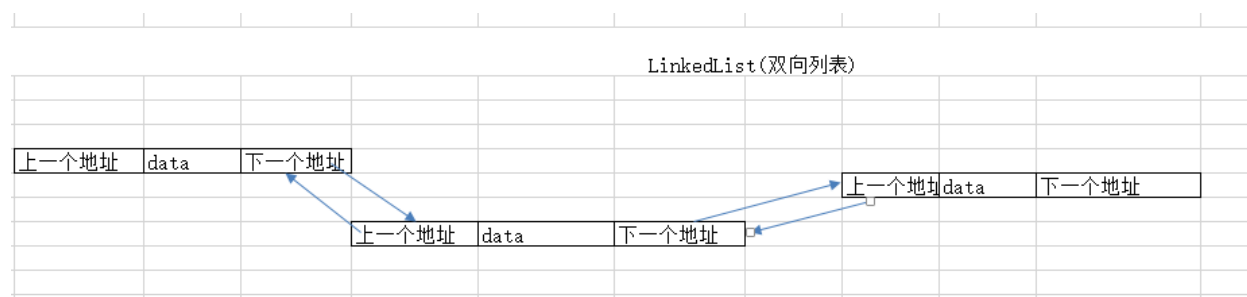
- ArrayList



- 单向链表



- 双向链表



4.2.常见命令

命令	命令描述
lpush key value1 value2...	将一个或多个值插入到列表头部(左边)
rpush key value1 value2...	在列表中添加一个或多个值(右边)
lpop key	左边弹出一个 相当于移除第一个
rpop key	右边弹出一个 相当于移除最后一个
llen key	返回指定key所对应的list中元素个数

5,存储set

5.1.概述

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。

Redis 中 集合是通过哈希表实现的，所以添加，删除，查找的时间复杂度都是O(1)。集合中最大的成员数为 $2^{32} - 1$ (4294967295, 每个集合可存储40多亿个成员)。

特点:无序+唯一

5.2.常见命令

命令	命令描述
sadd key member1 [member2]	向集合添加一个或多个成员
srem key member1 [member2]	移除一个成员或者多个成员
smembers key	返回集合中的所有成员,查看所有

6,Redis数据结构详解之Zset

Zset跟Set之间可以有并集运算，因为他们存储的数据字符串集合，不能有一样的成员出现在一个zset中，但是为什么有了set还要有zset呢？zset叫做有序集合，而set是无序的，zset怎么做到有序的呢？就是zset的每一个成员都有一个分数与之对应，并且分数是可以重复的。有序集合的增删改由于有啦排序，执行效率就是非常快速的，即便是访问集合中间的数据也是非常高效的。

命令	描述
zadd myZSet 1 zlh	添加分数为1，值为zlh的zset集合
zadd mySet 2 Tom 3 Jim	添加2条zset集合，分别为分数为2、3，值为tom、jim的集合
zcard myZSet	输出zset的成员个数为
zscore myZset tom	输出tom的分数为1
zrem myZSet Tom Jim	删除值为Tom和jim的两个zset成员
zrange mZySet 0 -1 [withscores]	0为开始，-1为结束,查看所有 也可以以角标查看,withscores按分数从小到大排序
zrevrange myZset 2 3 [withscores]	顺序反转,将zrange的顺序倒过来

五,Redis通用的操作和特性

1.通用操作

命令	描述
keys *	查询所有的key
exists key	判断是否有指定的key 若有返回1,否则返回0
expire key	秒数:设置这个key在缓存中的存活时间
ttl key	展示指定key的剩余时间,若返回值为 -1:永不过期,若返回值为 -2:已过期或者不存在
del key	删除指定key
rename key	新key:重命名
type key	判断一个key的类型
ping	测试连接是否连接

2.多数据库性

redis默认是16个数据库, 编号是从0~15.

命令	描述
select index	切换库
move key index	把key移动到几号库(index是库的编号)
flushdb	清空当前数据库
flushall	清空当前实例下所有的数据库

3.Redis在公司里面的应用和定位

- 问题
 - Redis可以取代mysql吗? 不可以
 - 公司里面能不能只有nosql数据(redis), 没有关系型数据(mysql,oracle)? 基本上是不行
- 定位
 - 作为一个补充产品, 用Redis做优化
- 使用
 - 先从关系型数据库,文件里面把数据读取出来存到redis里面, 下次再获得相同数据的时候就直接从Redis获得
- 哪些数据适合用Redis,哪些不适合Redis
 - 适合用的: 经常需要频繁获得的,但是又不经常改变的(eg: 省市, 类别的信息)
 - 不适合用的: 数据经常改变的, 对数据的精确度要求特别高的(eg: 账户信息)

六, Redis的持久化

1. Redis持久化概述

Redis的高性能是由于其将所有数据都存储在了内存中, 为了使Redis在重启之后仍能保证数据不丢失, 需要将数据从内存中同步到硬盘中, 这一过程就是持久化。

Redis支持两种方式的持久化, 一种是RDB方式, 一种是AOF方式。可以单独使用其中一种或将二者结合使用。

2.RDB持久化机制

2.1概述

RDB持久化是指在**指定的时间间隔内**将内存中的数据快照写入磁盘。这种方式就是将内存中数据以快照的方式写入到二进制文件中,默认的文件名为dump.rdb。 这种方式是默认已经**开启了,不需要配置**。

2.2 RDB持久化机制的配置

- 在redis.windows.conf配置文件中有如下配置：

```
98 save 900 1
99 save 300 10
100 save 60 10000
```

其中，上面配置的是RDB方式数据持久化时机：

关键字	时间(秒)	key修改数量	解释
save	900	1	每900秒(15分钟)至少有1个key发生变化，则dump内存快照
save	300	10	每300秒(5分钟)至少有10个key发生变化，则dump内存快照
save	60	10000	每60秒(1分钟)至少有10000个key发生变化，则dump内存快照

3. AOF持久化机制

3.1概述

AOF持久化机制会将每一个收到的写命令都通过write函数追加到文件中,默认的文件名是appendonly.aof。这种方式默认是没有开启的,要使用时候需要配置.

3.2AOF持久化机制配置

3.2.1开启配置

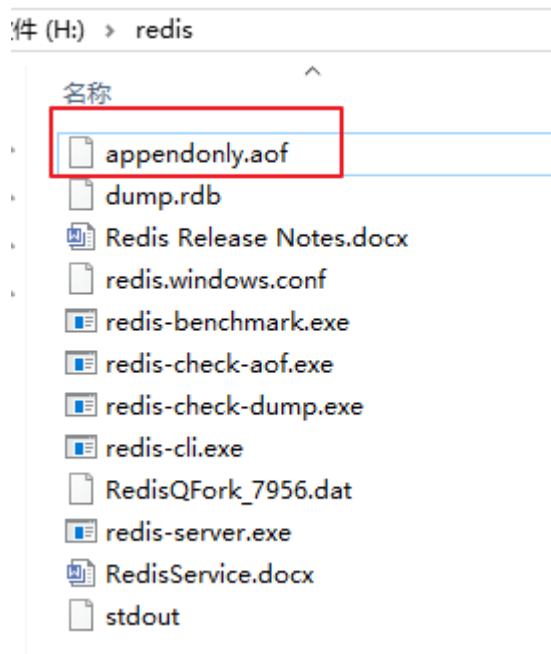
- 在redis.windows.conf配置文件中有如下配置：

```
391
392 appendonly no
```

- 将appendonly修改为yes，但是启动redis的时候需要指定该文件,也就是意味着不能直接点击了,需要输入命令启动:

```
redis-server.exe redis.windows.conf
```

- 开启aof持久化机制后，默认会在目录下产生一个appendonly.aof文件



3.2.2配置详解

- 上述配置为aof持久化的时机，解释如下：(在redis.windows.conf配置)

```
420 # appendfsync always
421 appendfsync everysec
422 # appendfsync no
```

关键字	持久化时机	解释
appendfsync	always	每执行一次更新命令，持久化一次
appendfsync	everysec	每秒钟持久化一次
appendfsync	no	不持久化

4,两种持久化机制比较

4.1RDB

优点

- RDB 是一个非常紧凑 (compact) 的文件，它保存了 Redis 在某个时间点上的数据集。这种文件非常适合用于进行备份
- RDB 在恢复大数据集时的速度比 AOF 的恢复速度要快(因为其文件要比AOF的小)
- RDB的性能更好

缺点

- RDB的持久化不够及时
- RDB持久化时如果文件过大可能会造成服务器的阻塞,停止客户端请求

4.2 AOF

优点

- AOF的持久性更加的耐久(可以每秒 或 每次操作保存一次)
- AOF 文件有序地保存了对数据库执行的所有写入操作， 这些写入操作以 Redis 协议的格式保存， 因此 AOF 文件的内容非常容易被别人读懂， 对文件进行分析（parse）也很轻松。
- AOF是增量操作

缺点

- 对于相同的数据集来说，AOF 文件的体积通常要大于 RDB 文件的体积
- 根据所使用的 fsync 策略，AOF 的速度可能会慢于 RDB。

4.3 选择

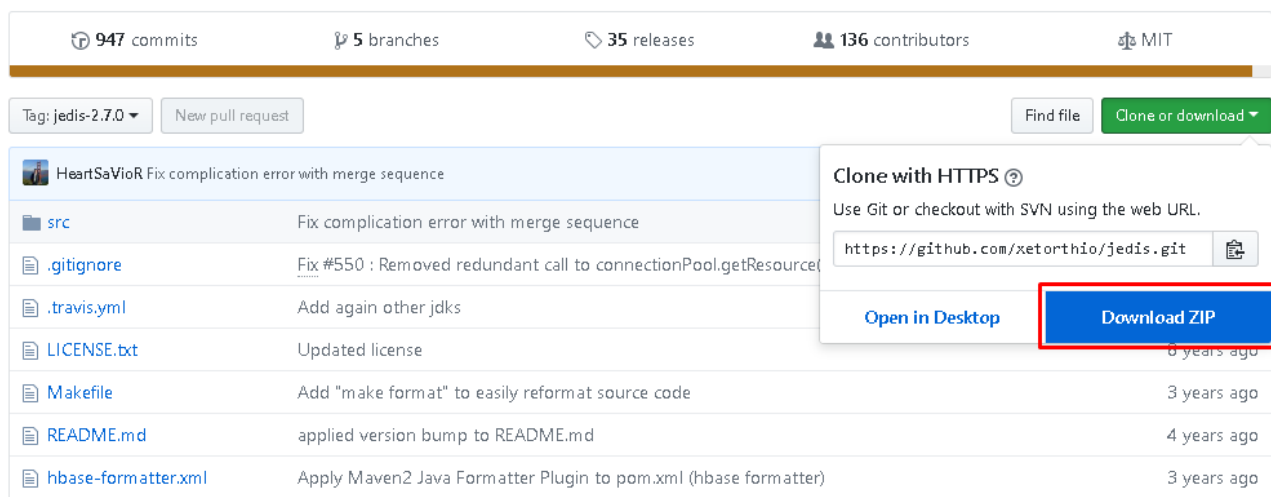
- 如果你非常关心你的数据， 但仍然可以承受数分钟以内的数据丢失， 选择RDB 持久化。
- 如果对数据的完整性要求比较高, 选择AOF

七, Jedis的基本使用

1.jedis的介绍

Redis不仅是使用命令来操作，现在基本上主流的语言都有客户端支持，比如java、C、C#、C++、php、Node.js、Go等。在官方网站里列一些java的客户端，有Jedis、Redisson、Jredis、JDBC-Redis、等其中官方推荐使用Jedis和Redisson。在企业中用的最多的就是Jedis，Jedis同样也是托管在github上，地址：<https://github.com/xetorthio/jedis>。

A blazingly small and sane redis java client



947 commits 5 branches 35 releases 136 contributors MIT

Tag: jedis-2.7.0 New pull request Find file Clone or download

HeartSaVioR Fix compilation error with merge sequence

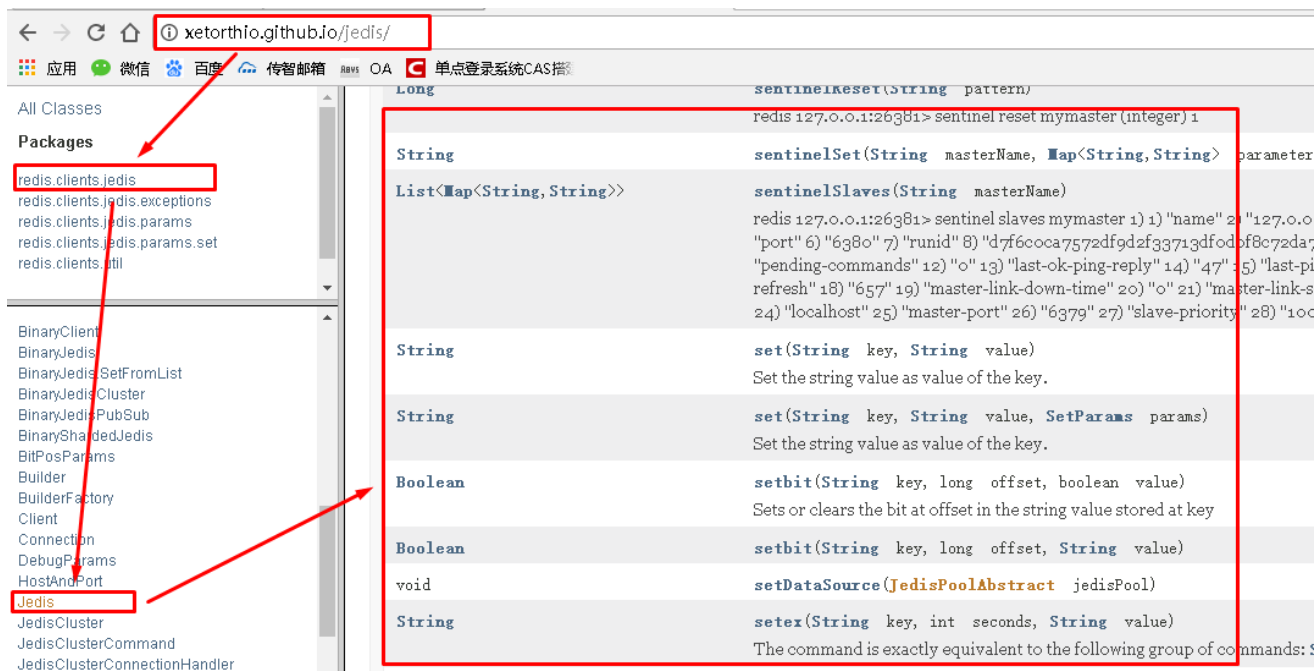
File	Description	Time
src	Fix compilation error with merge sequence	
.gitignore	Fix #550 : Removed redundant call to connectionPool.getResource	6 years ago
.travis.yml	Add again other jdks	
LICENSE.txt	Updated license	
Makefile	Add "make format" to easily reformat source code	3 years ago
README.md	applied version bump to README.md	4 years ago
hbase-formatter.xml	Apply Maven2 Java Formatter Plugin to pom.xml (hbase formatter)	3 years ago

Clone with HTTPS ⓘ
Use Git or checkout with SVN using the web URL.
<https://github.com/xetorthio/jedis.git>

Open in Desktop **Download ZIP**



文档地址:<http://xetorthio.github.io/jedis/>

- 官方API文档查询方式：



2.Jedis的使用

1. 导入jar包

 commons-pool2-2.3.jar	2015/4/18 13:11	Executable Jar File	107 KB
 jedis-2.7.0.jar	2015/4/18 13:11	Executable Jar File	332 KB

2. 创建jedis对象

```
new Jedis("192.168.17.136", 6379)
```

3. 操作redis数据库

4. 释放资源 close

3. jedis常用API

方法	解释
new Jedis(host, port)	创建jedis对象，参数host是redis服务器地址，参数port是redis服务端口
set(key,value)	设置字符串类型的数据
get(key)	获得字符串类型的数据
hset(key,field,value)	设置哈希类型的数据
hget(key,field)	获得哈希类型的数据
lpush(key,values)	设置列表类型的数据
lpop(key)	列表左面弹栈
rpop(key)	列表右面弹栈
del(key)	删除指定的key

4.jedis操作

- 基本操作

```
public class Di {
    public static void main(String[] args) {
        Jedis jedis=new Jedis("localhost",6379);
        jedis.set("MagicGirl","MyLove");
        jedis.close();
    }
}
```

5 jedis连接池的使用

5.1 jedis连接池的基本概念

jedis连接资源的创建与销毁是很消耗程序性能，所以jedis为我们提供了jedis的池化技术，jedisPool在创建时初始化一些连接资源存储到连接池中，使用jedis连接资源时不需要创建，而是从连接池中获取一个资源进行redis的操作，使用完毕后，不需要销毁该jedis连接资源，而是将该资源归还给连接池，供其他请求使用。

5.2jedis连接池的 使用

- 基本使用


```

public class Di {
    public static void main(String[] args) {
        JedisPoolConfig jpc=new JedisPoolConfig();
        jpc.setMaxTotal(10);
        JedisPool jp = new JedisPool(jpc,"localhost",6379);
        Jedis jedis = jp.getResource();
        jedis.set("wowo","redii");
        jedis.set("cienchen","low");
        System.out.println(jedis.get("wowo"));
        jedis.close();
        jp.close();
    }
}

```

- Jedis工具类的抽取(使用连接池)

```

public class JedisUtils {
    private static JedisPool jp;
    static{
        JedisPoolConfig jpc=new JedisPoolConfig();
        jpc.setMaxTotal(10);
        jp = new JedisPool(jpc,"localhost",6379);
    }
    public static Jedis getJ(){
        Jedis jedis = jp.getResource();
        return jedis;
    }
    public static void close(Jedis jedis){
        if(jedis != null){
            jedis.close();
        }
    }
}

```

- 优化后

```

public class Di {
    public static void main(String[] args) {
        Jedis jedis = JedisUtils.getJ();
        jedis.set("wowo","redii");
        jedis.set("cienchen","low");
        System.out.println(jedis.get("wowo"));
        JedisUtils.close(jedis);
    }
}

```

八,使用Redis优化省份的展示

1.案例需求

访问index.html页面，使用ajax请求加载省份列表，用户第一次访问数据库获取，以后都从redis里面获取。



2.思路分析

3.代码实现

3.1准备工作

- 数据库

```
CREATE TABLE `province` (  
  `pid` int NOT NULL AUTO_INCREMENT,  
  `pname` varchar(40) DEFAULT NULL,  
  PRIMARY KEY (`pid`)  
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8;  
  
INSERT INTO `province` VALUES ('1', '广东');  
INSERT INTO `province` VALUES ('2', '湖北');  
INSERT INTO `province` VALUES ('3', '湖南');  
INSERT INTO `province` VALUES ('4', '四川');  
INSERT INTO `province` VALUES ('5', '山东');  
INSERT INTO `province` VALUES ('6', '山西');  
INSERT INTO `province` VALUES ('7', '广西');
```

- Province.java

```
public class Province {  
  
    private Integer pid;  
    private String pname;  
    get/set方法  
    toString方法  
}
```

- 页面

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>  
<html>  
  <head>  
    <title>$Title$</title>  
  
    <script src="jquery-3.3.1.min.js"></script>
```

```

</head>
<body>
省份
<select id="shengxuan" name="province">
    <option value="-1">请选择</option>

</select>
</body>
<script>
$.post("/ms",function (result) {
    $(result).each(function (i,obj) {
        $("#shengxuan").append("<option value='"+obj.pid+"'>"+obj.pname+"</option>");
    })
}, "json");
</script>
</html>

```

- Dao

```

public class Dao {
public List getlist()throws Exception{
    JdbcTemplate jt=new JdbcTemplate(C3P0uTILS.getS());
    List<Province> list = jt.query("select * from province",new BeanPropertyRowMapper<>
(Province.class));
    return list;
}
}

```

- Service

```

public class Service {
    public String getP() throws Exception {
        String data=null;
        Jedis jedis=null;
        try {
            jedis=JedisUtils.getJ();
            data=getjedisre(jedis);
            if(data==null){
                data=getMysql();
                savei(jedis,data);
            }

        } catch (Exception e) {
            System.out.println("redis服务器异常");
            data=getMysql();
            e.printStackTrace();
        }finally {
            JedisUtils.close(jedis);
        }
        return data;
    }
}

```

```

private void save(Jedis jedis, String data) {
    if(jedis!=null){
        jedis.set("province_key",data);
    }
}

private String getJedisre(Jedis jedis) {
    if(jedis!=null){
        return jedis.get("province_key");
    }
    return null;
}

public String getMysql() throws Exception {
    Dao dao=new Dao();
    List<Province> list = dao.getList();
    ObjectMapper objectMapper = new ObjectMapper();
    String s = objectMapper.writeValueAsString(list);
    return s;
}
}

```

- Servlet

```

@WebServlet("/ms")
public class MyServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        try {
            response.setContentType("text/html;charset=utf-8");
            Service ss=new Service();
            String data = ss.getP();
            response.getWriter().print(data);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

- JedisUtils

```

public class JedisUtils {
    private static JedisPool jp;
    static{
        JedisPoolConfig jpc=new JedisPoolConfig();

        jpc.setMaxTotal(10);
    }
}

```

```
        jp = new JedisPool(jpc,"localhost",6379);
    }
    public static Jedis getJ(){
        Jedis jedis = jp.getResource();
        return jedis;
    }
    public static void close(Jedis jedis){
        if(jedis != null){
            jedis.close();
        }
    }
}
```