day31-Redis

学习目标

- [ ] 能够说出redis的常用数据类型
- [ ] 能够使用redis的string操作命令
- [ ] 能够使用redis的hash操作命令
- [ ] 能够使用redis的list操作命令
- [ ] 能够使用redis的set操作命令
- [ ] 能够使用redis的zset操作命令
- [ ] 能够使用jedis操作Redis
- [ ] 能够理解Redis持久化

第一章-Redis介绍和安装

知识点-Nosql概述

1.目标

- 能够理解nosql的概念

2.路径

- 什么是NOSQL
- 为什么需要NOSQL
- 主流的NOSQL产品
- NOSQL的特点

3.讲解

3.1 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：关系型数据库，按行存储，每行表示一个实体(对象)。Oracle、SQLServer...

3.2.为什么需要学习NOSQL （三高）

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(关系型)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数据库应运而上，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。(微博： 明星曝光恋情 、 离婚)

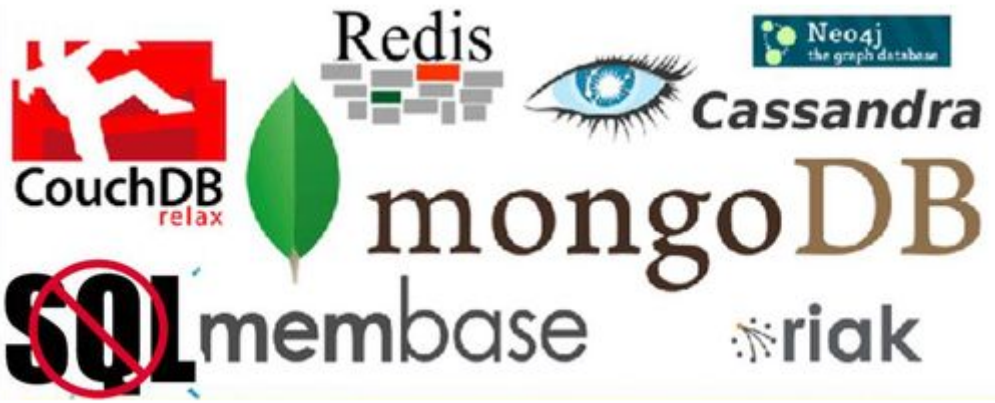
- Huge Storage - 对海量数据的高效率存储和访问的需求

  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?

3.3 主流的NOSQL产品



- 键值(Key-Value)存储数据库 redis
- 列存储数据库(分布式) Hbase
- 文档型数据库 (Web应用与Key-Value类似，Value是结构化的) mongodb
- 图形(Graph)数据库(图结构) facebook 微信 社交圈子 发朋友圈

3.4NOSQL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。

4.小结

1. NoSql：非关系型数据库
2. 为什么要学习NoSQL
    - 高并发读写
    - 海量数据查询效率
    - 高扩展, 高可用

    > 解决关系型数据库的瓶颈，它是关系型数据库的一个补充。

3. NoSQL产品
    - Redis
    - MongoDB

知识点-Redis概述

1.目标

- 知道什么是Redis以及Redis的应用场景

2.路径

- 什么是Redis
- redis的应用场景

3.讲解

3.1.什么是Redis 内存数据库 按照键值对进行数据存储

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，**数据是保存在内存里面的**. 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支持的值数据类型如下：

key：string

value 5种：string 、hash 、list、set、sortedset

- 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

3.2 redis的应用场景

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

4.小结

1. Redis:是由C语言开发的一个NoSQL数据库(==内存==数据库)，以==key-value键值对==存储，读写效率高。
2. Redis应用场景
    - 缓存(商品、首页内容)
    - 队列(秒杀)
    - 数据过期处理(短信验证码)
    - 分布式集群架构 session共享

实操-window版Redis的安装

1.目标

- 掌握window版Redis的安装与使用

2.路径

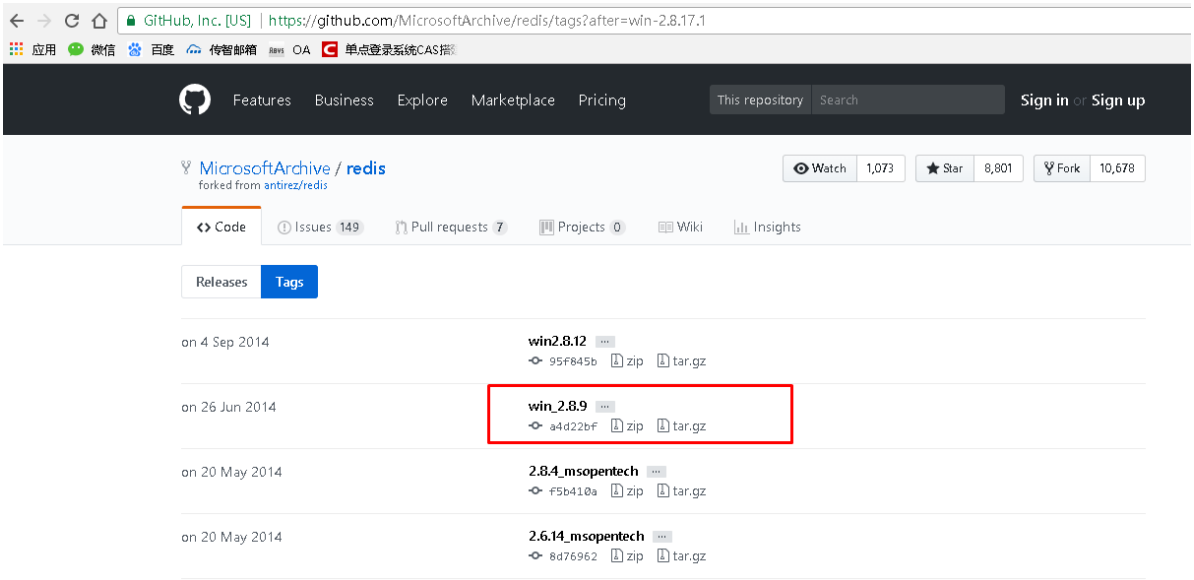
1. Redis的下载
2. Redis的安装
3. 启动

3.讲解

3.1.windows版Redis的下载

官方提倡使用Linux版的Redis，所以官网只提供了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

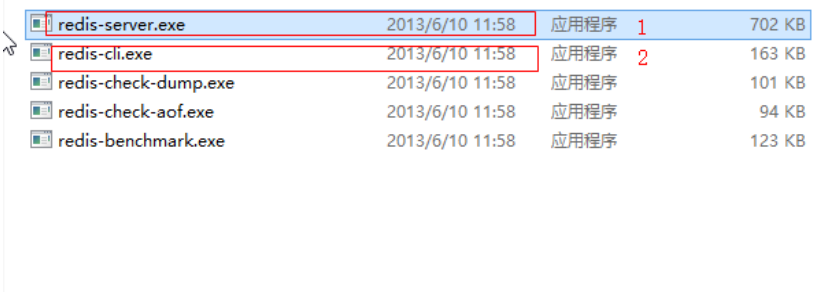
3.2 Redis的安装

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**

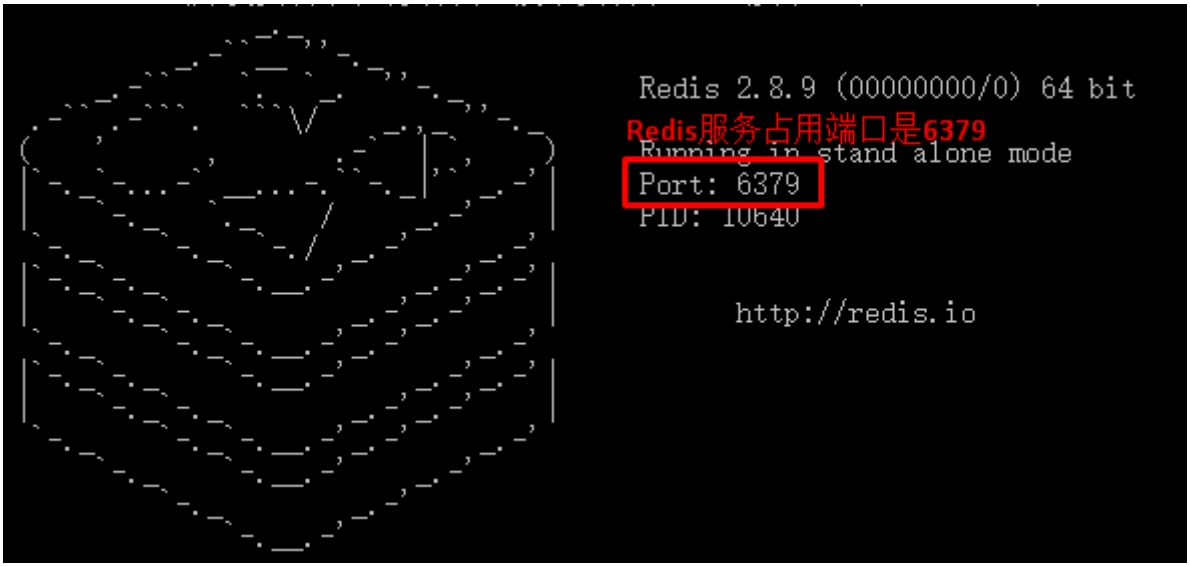
3.3 启动

先启动服务端，再启动客户端！



- 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**

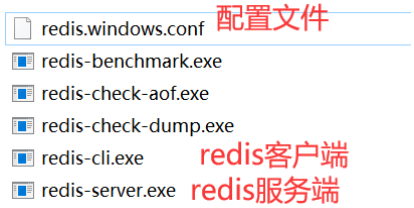
![](

- 点击redis-cli

)

4.小结

1. redis目录结构



2. 启动
    - 先点击redis-server.exe
    - 再点击redis-cli.exe
3. Redis
    - 端口是 6379
    - 默认不需要密码

实操-Linux版本Redis的安装

1.目标

- [ ] 掌握Redis的安装

2.讲解

1. 在Linux虚拟机中安装c++环境(C的编译运行环境)

```
# 注意：由于CentOS6.x版本已经相对较老，官方不在维护其yum安装的软件源，需要替换yum安装软件源仓库
# https://blog.csdn.net/qq_32279165/article/details/110957782
yum -y install gcc-c++
```

2. 下载Redis
3. 上传到Linux
4. 解压

```
mkdir /usr/local/redis

tar -zxvf /root/redis-4.0.14.tar.gz -C /usr/local/redis
```

5. 编译

```
cd redis-4.0.14
make
```

6. 安装

```
make install PREFIX=/usr/local/redis
```

7. 进入安装好的redis目录,复制配置文件

```
cd /usr/local/redis/bin
cp /root/redis-4.0.14/redis.conf ./
```

8. 修改配置文件

```
# 修改配置文件
vi redis.conf
# Redis后台启动
修改 daemonize 为 yes
# Redis服务器可以跨网络访问
修改 bind 为 0.0.0.0
```

9. 启动redis

```
./redis-server redis.conf
```

3.小结

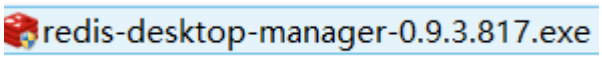
1. 对着笔记装就可以!

实操-Redis的客户端安装

1.目标

装Redis的客户端

2.步骤



- 双击, 下一步 ...
- 安装没有中文和空格目录

第二章-Redis的数据类型【重点】

知识点-redis中数据结构/类型

1.目标

- 能够说出Redis中的数据类型

2.路径

- Redis的数据类型类型介绍
- Redis中的key

3.讲解

3.1Redis的数据类型【面试】

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)　　【使用最多】

哈希(hash)　　java中map

字符串列表(list)　java中LinkedList　链表存储

字符串集合(set)　java中set hashset

有序的字符串集合(sorted-set或者叫zset)　　有序+唯一

3.2key

- key不要太长(不能>1024个字节)
- 也不要太短 . 可读性差.

```
项目名_模块_key  eg: jd_user_name
项目名:模块:key  eg：jd:user:name
```

4.小结

1. Redis数据类型
    - string 【最多】
    - hash
    - list
    - set
    - zset
2. 键的命名

```
项目_模块_key
项目:模块:key
```

知识点-Redis字符串(String)

1.目标

- 能够使用redis的string操作命令

2.路径

1. Redis字符串(String)概述
2. 应用场景
3. 常见命令
4. 应用举例

2.讲解

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应用场景

1. 缓存功能：字符串最经典的使用场景，redis作为缓存层，MySQL作为存储层，绝大部分请求数据都是在redis中操作，由于redis具有支撑高并发特性，所以缓存通常能起到加速读写和降低 后端压力的作用。
2. 计数器功能：比如视频播放次数，点赞次数。

2.3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**
GETSET key value	将给定 key 的值设为 value ，并返回 key 的旧值(old value)。
SETEX key seconds value	**将值 value 关联到 key ，并将 key 的过期时间设为 seconds (以秒为单位)。**
SETNX key value	**只有在 key 不存在时设置 key 的值。**
INCR key	**将 key 中储存的数字值增一。**
INCRBY key increment	将 key 所储存的值加上给定的增量值（increment） 。
DECR key	将 key 中储存的数字值减一。
DECRBY key decrement	key 所储存的值减去给定的减量值（decrement） 。

2.4应用举例

商品编号、订单号采用string的递增数字特性生成。

```
定义商品编号key：items:id
192.168.101.3:7003> INCR items:id
(integer) 2
192.168.101.3:7003> INCR items:id
(integer) 3
```

4.小结

4.1String

1. String是redis中用的最多的一个数据类型，value最大可以存储512M的数据。
2. 我们可以把java对象转成json 字符串再存进去
3. 应用
    - 缓存
    - 递增,减(点赞...)
    - 过期时间处理
    - 分布式锁...

4.2使用string的问题

假设有User对象以JSON序列化的形式存储到Redis中，User对象有id，username、password、age、name等属性，存储的过程如下： 保存、更新： User对象 ==> json(string) ==>redis

如果业务只想更新用户年龄age，这个时候按正常，就需要取出json字符串，转为user对象，重新设置age属性值，然后再将User对象==> json(string) ==>redis ，优化？hash存储 就可以直接修改age即可 不需要转来转去

知识点-Redis 哈希(Hash)

1.目标

- 能够使用redis的hash操作命令

2.路径

1. Redis 哈希(Hash)概述
2. 应用场景
3. 常见命令
4. 应用举例

3.讲解

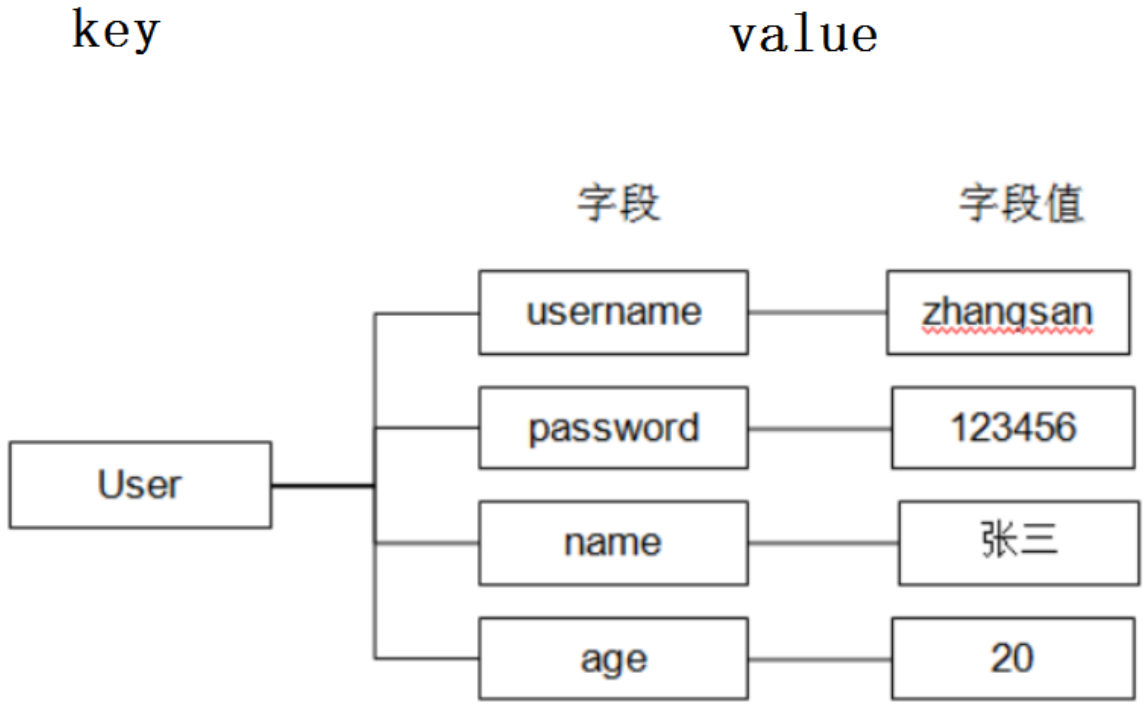
3.1 概述

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==.

hash特别适合存储对象相比较而言，**将一个对象类型存储在Hash类型里要比存储在String类型里占用更少的内存空间并方便存取整个对象**



3.2应用场景

用一个对象来存储用户信息，商品信息，订单信息等等。 **存储对象**

3.3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)
HGETALL key	获取在哈希表中指定 key 的所有字段和值
HKEYS key	获取所有哈希表中的字段
HVALS key	获取哈希表中所有值

3.4应用举例 存对象

存储商品信息

- 商品字段【商品id、商品名称、商品价格】
- 定义商品信息的key, 商品1001的信息在 Redis中的key为：[items:1001]
- 存储商品信息

```
HMSET items:1001 id 3 name apple price 999.9
```

4.小结

1. Redis是键值对存储，键key只能是String类型，值Value可以是String，hash(Map)、list、set、zse类型
2. Redis值存储使用hash类型特别适合存储对象，且方便进行操作

知识点-Redis 列表(List)

1.目标

- [ ] 能够使用redis的list操作命令

2.路径

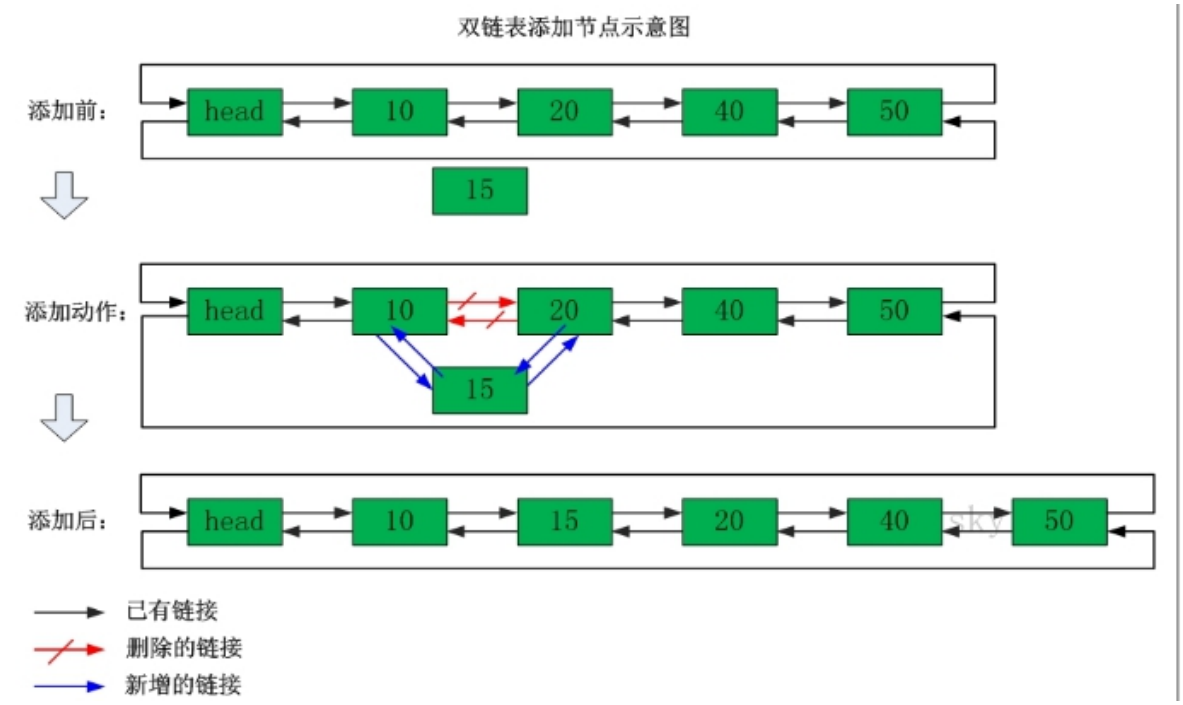
1. List类型
2. Redis 列表(List)概述
3. 应用场景
4. 常见命令
5. 应用举例

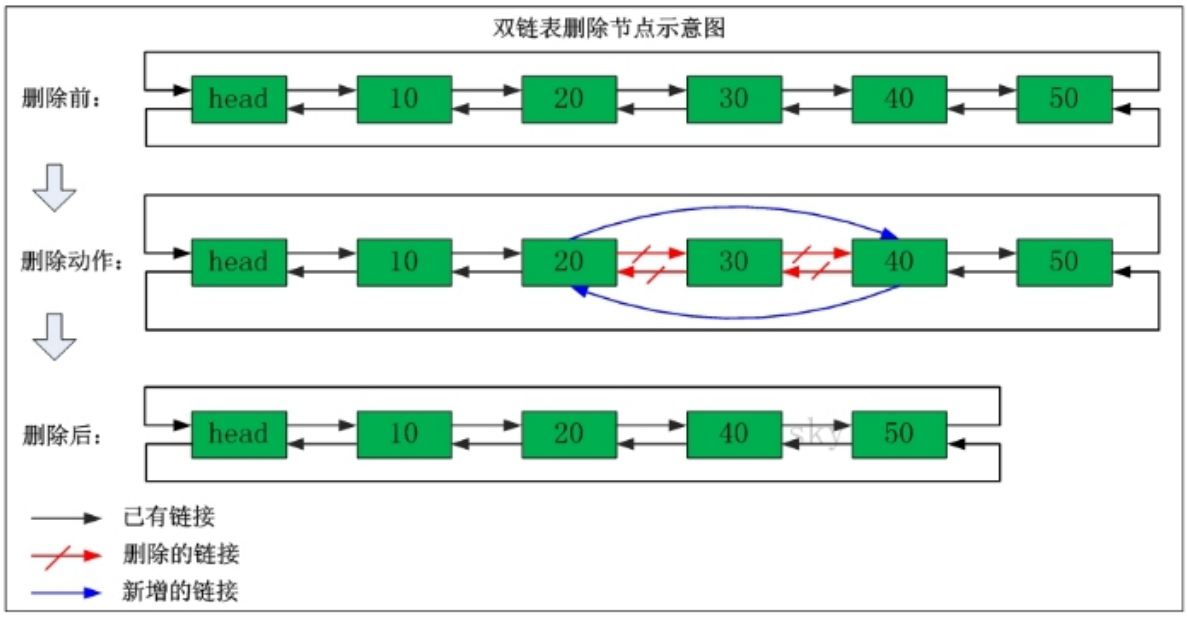
3.讲解

3.1List类型

ArrayList使用数组方式存储数据，所以根据索引查询数据速度快，而新增或者删除元素时需要设计到位移操作，所以比较慢。

LinkedList使用双向链表方式存储数据，每个元素都记录前后元素的指针，所以插入、删除数据时只是更改前后元素的指针指向即可，速度非常快。然后通过下标查询元素时需要从头开始索引，所以比较慢，但是如果查询前几个元素或后几个元素速度比较快。





3.2概述

列表类型（list）可以存储一个`有序的字符串列表(链表)`，常用的操作是向列表两端添加元素，或者获得列表的某一个片段。

列表类型内部是使用**双向循环链表（double linked list）**实现的，所以向列表两端添加元素的**时间复杂度为O(1)**，获取越接近两端的元素速度就越快。这意味着即使是一个有几千万个元素的列表，获取头部或尾部的10条记录也是极快的。

> eg：现在老师布置了一个作业，要求张三和李四去设计程序完成，张三写的程序运行时间50ms，占用空间5000M，李四写的程序运行时间50s，占用空间500M。
>
> 衡量一个程序是否合格高效，主要有两个维度：运行时间(程序的效率)、运行空间
>
> 时间复杂度：衡量程序运行是否高效
>
> 空间复杂度：衡量程序运行占用空间大小

> java程序：空间换时间，宁肯牺牲空间(+内存 +主机)，也要保证时间
>
> https://blog.csdn.net/qq_41523096/article/details/82142747

3.2应用场景

如好友列表，粉丝列表，消息队列，最新消息排行等。

rpush方法就相当于将消息放入到队列中，lpop/rpop就相当于从队列中拿去消息进行消费

3.3.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
LINDEX key index	通过索引获取列表中的元素
LINSERT key BEFORE| AFTER pivot value	在列表元素前或后插入元素 `eg: linsert words before c e`

3.4应用举例

商品评论列表

- 思路: 在Redis中创建商品评论列表,用户发布商品评论，将评论信息转成json存储到list中。用户在页面查询评论列表，从redis中取出json数据展示到页面。

4.小结

1. List数据结构是一个双向循环列表
2. 在进行list类型添加元素时，如果key存在，进行添加，如果key不存在，直接新建添加。

知识点-Redis 集合(Set)

1.目标

- 能够使用redis的set操作命令

2.路径

1. Redis 集合(Set)概述
2. 应用场景
3. 常见命令
4. 应用举例

3.讲解

3.1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的32次方 -1 (4294967295, 每个集合可存储40多亿个成员)。

Redis还提供了多个集合之间的交集、并集、差集的运算

特点:无序+唯一

3.2应用场景

投票记录

共同好友、共同兴趣、分类标签

3.3.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
SCARD key	获取集合的成员数
SPOP key	返回集合中移除的一个随机元素
SDIFF key1 [key2]	返回给定所有集合的差集
SUNION key1 [key2]	返回所有给定集合的并集
SINTER key1 [key2]	返回给定所有集合的交集

3.4应用举例

共同好友

- A的好友
- B的好友
- A和B的共同好友 SINTER

4.小结

1. Redis里面的Set 无序+唯一的, 类似Java里面的HashSet
2. 应用
    - 投票的记录
    - 求差值
    - 共同好友(求交集)

知识点-Redis 有序集合(sorted set | zset)

1.目标

- 能够使用redis的sorted set操作命令

2.路径

1. sorted set介绍
2. sorted set 应用场景
3. sorted set 命令
4. sorted set 具体示例

3.讲解

3.1概述

Redis 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。**

集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是O(1)。 集合中最大的成员数为 232 - 1 (4294967295, 每个集合可存储40多亿个成员)。

特点: 有序(根据分数排序)+唯一

3.2应用场景

排行榜：例如音乐排行榜 今年最流行十首华语歌曲

3.3.常见命令

命令	命令描述
ZADD key score member [score member ...]	增加元素
ZSCORE key member	获取元素的分数
ZREM key member [member ...]	删除元素
ZCARD key	获得集合中元素的数量
ZRANGE key start stop[WITHSCORES]	获得排名在某个范围的元素列表 查看所有元素：zrange key 0 -1 倒序排列查看：zrevrange key 0 2

3.4应用举例

商品销售排行榜

- 需求：根据商品销售量对商品进行排行显示
- 思路：定义商品销售排行榜（sorted set集合），Key为items:sellsort，分数为商品销售量
- 实现

```
--商品编号1001的销量是9，商品编号1002的销量是10 ，商品编号1003的销量是15
127.0.0.1:6379> zadd items:sellsort 9 1001 10 1002 15 1003
--商品编号1001的销量加1
127.0.0.1:6379> zincrby items:sellsort 1 1001
--商品销量前10名
127.0.0.1:6379> zrevrange items:sellsort 0 9
```

4.小结

1. ZSET：有序集合(有序+唯一)，顺序通过元素附带的score实现，注意：元素依然不能重复，但是score可以重复。
2. 使用场合：排行榜

第三章-Redis通用的操作,发布订阅和持久化

知识点-Redis通用的操作

1.目标

- 掌握Redis通用的操作命令

2.路径

- 通用操作
- 多数据库性

3.讲解

3.1 通用操作

- 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

3.2多数据库性

redis默认是16个数据库, 编号是从0~15. 【默认是0号库】

- 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

4.小结

1. ==exists key== 判断是否有这个key
2. ==expire key 秒数== 设置这个key在缓存中的存活时间
3. ==ttl key== 展示指定key的剩余时间(-1：永不过期 -2：已过期或不存在)
4. ==select index== 切换库

知识点-订阅发布机制 【了解】

1.目标

- [ ] 了解Redis订阅发布机制

2.路径

1. 什么是Redis订阅发布机制
2. 相关的命令
3. 订阅发布实操

3.讲解

3.1什么是Redis订阅发布机制

Redis 发布订阅(pub/sub)是进程间一种消息通信模式：发送者(pub)发送消息，订阅者(sub)接收消息。(用户收听广播)

Redis 客户端可以订阅任意数量的频道。

3.2相关的命令

序号	命令及描述
1	PUBLISH channel message 将信息发送到指定的频道。
2	SUBSCRIBE channel [channel ...] 订阅给定的一个或多个频道的信息。
3	UNSUBSCRIBE [channel [channel ...]] 指退订给定的频道

3.3订阅发布实操

- 开启两个客户端
- A客户端

```
SUBSCRIBE nba
```

- B客户端

```
PUBLISH  nba aaa
```

4.小结

1. Redis 发布订阅(pub/sub)是进程间一种消息通信模式, **工作里面一般使用MQ【消息中间件 rabbitMQ kafaka RockectMQ】**

知识点-Redis的持久化【面试】

Redis将数据保存在内存中，将内存中的数据保存到硬盘上，这个过程就是持久化。 为了数据备份

1.目标

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(文件)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【默认】**，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.路径

- RDB持久化机制
- AOF持久化机制
- 两种持久化机制比较

3.讲解

3.1RDB持久化机制

3.1.1概述

RDB持久化是指在==**指定的时间间隔内**==将内存中的数据集快照写入磁盘。这种方式是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**.==

3.1.2 RDB持久化机制的配置

- 在redis.windows.conf配置文件中有如下配置【持久化策略】:

```
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

3.2 AOF持久化机制

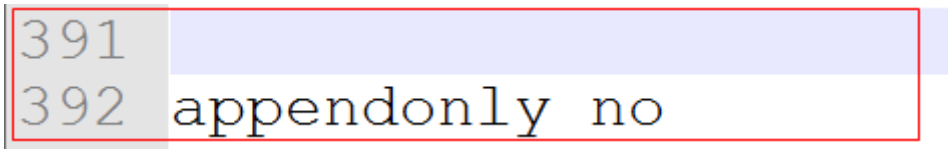
3.2.1概述

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.

3.2.2AOF持久化机制配置

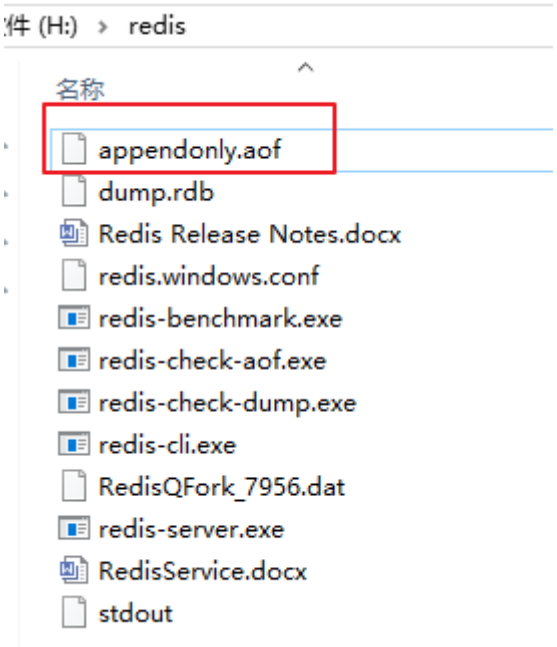
3.2.2.1开启配置

- 在redis.windows.conf配置文件中有如下配置:



- 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:

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



3.2.2.2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,小结

Redis持久化有两种方式：RDB【默认 二进制文件 有一定的时间间隔】、AOF【需要打开 命令文件 默认每秒】

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的性能要比AOF更好**==

缺点

- ==**RDB的持久化不够及时(一定时间间隔),可能会存在数据丢失**==
- 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

4.2AOF

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) 数据完整性较好**==
- 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

> RDB：默认开启　在一定的时间间隔内进行持久化 持久化生成的二进制文件 文件体积较小　恢复数据较快　但是容易发生数据丢失现象
> AOF：需要手动开启，可以设置每秒或每次操作都进行持久化 是一个增量文件 保存的是redis操作命令 文件易读易分析 体积较大 恢复数据较慢 不容易出现数据丢失现象 对于数据的完整性保存较好。

第四章-Jedis 【重点】

案例-Jedis的快速入门

1.目标

- [ ] 掌握什么是Jedis

2.路径

1. jedis的介绍
2. Jedis的入门

3.讲解

3.1 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、lettuce。 在企业中用的最多的就是Jedis，Jedis同样也是托管在github上.

==说白了**Jedis就是使用Java操作Redis的客户端(工具包) jedis=JDBC+驱动** ==

地址：https://github.com/xetorthio/jedis。

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

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
sadd(String key, String... members)	设置set类型的数据
zrange(String key, long start, long end)	获得在某个范围的元素列表
exists(key)	判断一个key是否存在
del(key)	删除key

3.2 Jedis的入门

需求: 使用java代码操作Redis 进行增(改)删查

步骤:

1. 创建项目 添加jar包
2. 创建jedis对象
3. 调用方法
4. 关闭对象 释放资源

- 基本操作

```
package com.itheima.demo;

import redis.clients.jedis.Jedis;

import java.util.Set;

/**
 * Jedis快速入门使用
 */
public class JedisDemo01 {
    public static void main(String[] args) {
        //0.导入jedis jar包
        //1.创建jedis对象
        Jedis jedis = new Jedis("127.0.0.1", 6379);
        //2.调用方法
        //2.1:string类型操作
        /*jedis.set("akey","aaa");
```

```
        String akey = jedis.get("akey");
        System.out.println("akey = " + akey);
        Boolean flag = jedis.exists("bkey");
        System.out.println("flag = " + flag);
        jedis.del("akey");*/
        //2.hash类型操作
        /*jedis.hset("user","name","zs");
        System.out.println("user-name = " + jedis.hget("user", "name"));*/
        //3.list类型操作
        /*jedis.rpush("list","a","b","c","d","e");
        System.out.println("lpop = " + jedis.lpop("list"));
        System.out.println("rpop = " + jedis.rpop("list"));*/
        //4.set类型操作
        /*jedis.sadd("set","中国","美国","小日本");
        System.out.println("set-length = " + jedis.scard("set"));*/
        //5.zset类型操作
        jedis.zadd("s1",88,"zs");
        jedis.zadd("s1",99,"ls");
        jedis.zadd("s1",100,"ww");

        Set<String> s1 = jedis.zrevrange("s1", 0, 2);
        System.out.println(s1);

        //3.关闭对象 释放资源
        jedis.close();
    }
}
```

4.小结

1. Jedis: java操作Redis的客户端, 工具包(**等价于 JDBC+数据库驱动**)，操作redis，也就是**把之前redis的命令封装成了方法。**
2. 使用步骤
    - 创建项目 导入jar包
    - 创建jedis对象
    - 调用方法
    - 关闭对象 释放资源

知识点-Jedis进阶

1.目标

- [ ] 掌握Jedis连接池的使用

2.路径

1. jedis连接池介绍
2. jedis连接池的使用
3. 工具类的抽取

3.讲解

3.1 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。

3.2jedis连接池的使用

需求: 从Jedis的连接池里面获得jedis

步骤:

1. 创建JedisPoolConfig配置对象(配置数据源)
2. 创建JedisPool对象
3. 获取Jedis对象
4. 调用方法操作
5. 关闭对象 释放资源 归还到连接池

- 基本使用

```
package com.itheima.demo;

import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisPool;
import redis.clients.jedis.JedisPoolConfig;

public class JedisPoolDemo02 {
    public static void main(String[] args) {
        //1.创建jedisPoolconfig对象    设置数据源
        JedisPoolConfig jedisPoolConfig = new JedisPoolConfig();
        jedisPoolConfig.setMaxTotal(10);
        jedisPoolConfig.setMaxWaitMillis(3000);
        //2.创建jedispool对象
        JedisPool jedisPool = new JedisPool(jedisPoolConfig, "localhost", 6379);
        //3.获得jedis对象
        Jedis jedis = jedisPool.getResource();
        //4.调用方法
        Boolean flag = jedis.exists("akey");
        if(flag){
            System.out.println("akey存在："+jedis.get("akey"));
        }else{
            System.out.println("不存在");
            jedis.set("akey","aaa");
        }
        //5.将jedis对象归还到连接池
        jedis.close();
    }
}
```

3.3Jedis工具类的抽取

目的: 1.保证池子只有一个 2.获得jedis对象 3.归还

步骤:

1. 创建JedisUtils类
2. 在类中使用静态代码块初始化JedisPool连接池对象
3. 提供获取jedis对象方法 getJedis()
4. 提供归还方法 close()

```
package com.itheima.utils;

import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisPool;
import redis.clients.jedis.JedisPoolConfig;

/**
 * jedis工具类:
 *  1.抽取获取jedis对象的代码  封装成一个方法  getJedis
 *  2.抽取jedis对象归还的代码  封装成一个方法  close(Jedis jedis)
 *  问题:
 *      每次获取jedis对象都要创建一个jedisPool对象
 *  优化:
 *      使用单例模式，通过静态代码块进行代码优化
 *
 */
public class JedisUtils {
    private static JedisPool jedisPool;

    static {
        //1.创建jedisPoolconfig对象    设置数据源
        JedisPoolConfig jedisPoolConfig = new JedisPoolConfig();
        jedisPoolConfig.setMaxTotal(10);
        jedisPoolConfig.setMaxWaitMillis(3000);
        //2.创建jedispool对象
        jedisPool = new JedisPool(jedisPoolConfig, "localhost", 6379);
    }

    /**
     * 获取jedis对象
     * @return
     */
    public static Jedis getJedis(){

        //3.获得jedis对象
        return jedisPool.getResource();
    }

    public static void close(Jedis jedis){
        if(jedis!=null){
            jedis.close();
        }
    }
}
```

3.4.3 作业

> 把JedisUtils里面配置信息(eg：host和port) 抽取到 jedis.properties里面

4.小结

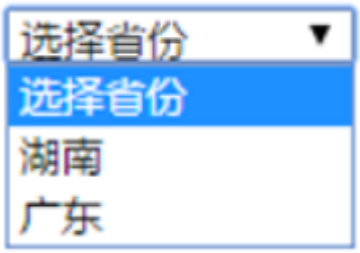
1. 使用JedisPool目的: 使用池化技术，避免jedis对象频繁的创建销毁造成系统的资源浪费！

案例-使用Redis优化省份的展示

1.需求

访问index.html页面，使用ajax请求加载省份列表(响应json)　vue的axios

- 先从Redis里面获得
  - 有 就直接返回
  - 没有 从Mysql获得,再存到Redis



2.分析

2.1直接从MySQL获得

1. 创建项目，准备数据库、页面，添加jar包、工具类C3P0Utils、配置文件c3p0.xml
2. 在index.html页面发起ajax请求，获得后台响应的省份列表json数据 填充到页面展示

```
//页面一加载 就请求获得数据
created:function(){
    //使用axios发送ajax请求 获取数据，进行绑定
    axios.get(url).then(response=>{
        //进行判断 数据绑定
    });
}
```

3. 编写Servlet　ProvinceServlet

```
//1.判断redis是否存在省份列表数据
jedis.exists("province_list");
//2.判断结果 没有
    //2.1：从MySQL数据库中查询
    //2.2：将查询出来的省份列表数据 存入到redis
    //2.3：将省份列表数据响应到前端页面
//2.判断结果 redis中有
    //2.1：根据key获取到
    //2.2：将省份列表数据以json个数响应到前端页面
```

4. 编写Service　ProvinceService

5. 编写Dao　　ProvinceDao

3.代码实现

3.1准备工作

- 数据库

```
create database day31;
use day31;

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');
```

- 创建工程(web)
- 导入jar包, 导入配置文件, 导入工具类,导入页面
- Province.java

```
public class Province {

    private Integer pid;
    private String pname;
    public Integer getPid() {
        return pid;
    }
    public void setPid(Integer pid) {
        this.pid = pid;
    }
    public String getPname() {
        return pname;
    }
    public void setPname(String pname) {
        this.pname = pname;
    }
    @Override
    public String toString() {
        return "Province [pid=" + pid + ", pname=" + pname + "]";
    }
}
```

3.2代码实现

- 页面

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <div id="app">
        省: <select id="pSelect">
                  <option>请选择</option>
                  <option v-for="(province,index) in provinces">
{{province.pname}}</option>
           </select>
    </div>

    <script src="js/vuejs-2.5.16.js"></script>
    <script src="js/axios-0.18.0.js"></script>
    <script>
        var vm = new Vue({
            el:"#app",
            data:{
                provinces:[]
            },
            methods:{

            },
            created:function () {
                axios.get("province").then(response=>{
                    if(response.data.flag){
                        //数据绑定
                        this.provinces = response.data.result;
                    }else{
                        alert(response.data.message);
                    }
                });
            }
        });
    </script>
</body>
</html>
```

- ProvinceServlet

```
package com.itheima.web;

import com.alibaba.fastjson.JSON;
import com.itheima.bean.Province;
import com.itheima.bean.Result;
import com.itheima.service.ProvinceService;
import com.itheima.utils.JedisUtils;
import com.itheima.utils.JsonUtils;
```

```
import jdk.nashorn.internal.scripts.JS;
import redis.clients.jedis.Jedis;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

@WebServlet(value = "/provinceServlet")
public class ProvinceServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        Jedis jedis = null;
        try {
            //1.获取请求参数
            //2.调用业务处理

            //2.1.判断redis是否存在省份列表数据
             jedis= JedisUtils.getJedis();
            Boolean flag = jedis.exists("province_list");
            if(flag){
                System.out.println("从redis中查询...");
                //2.判断结果 redis中有
                //2.1：根据key获取到    从redis中获取到的数据不是json字符串 只是普通字符串
所以响应给前台 前台无法正常解析!
                String province_list = jedis.get("province_list");
                //注意：将字符串pr_list转为省份list集合数据
                List<Province> list = (List<Province>)
JSON.parse(province_list);
                //2.2：将省份列表数据以json个数响应到前端页面
                //3.1:封装统一响应体
                Result result = new Result(true, "查询成功", list);
                //3.2：使用工具类向前端响应json数据
                JsonUtils.printResult(response,result);
            }else{
                System.out.println("从MySQL中查询...");
                //2.判断结果 没有
                //2.1：从MySQL数据库中查询
                //2.2：将查询出来的省份列表数据 存入到redis
                ProvinceService provinceService = new ProvinceService();
                //获取所有的省份列表
                List<Province> list = provinceService.findAll();
                //注意1：需要将list集合数据转json字符串存入redis
                jedis.set("province_list", JSON.toJSONString(list));

                //2.3：将省份列表数据响应到前端页面
                //3.响应
                //3.1:封装统一响应体
                Result result = new Result(true, "查询成功", list);
                //3.2：使用工具类向前端响应json数据
                JsonUtils.printResult(response,result);
            }
        } catch (Exception e) {
```

```
            e.printStackTrace();
            //3.1:封装统一响应体
            Result result = new Result(false, "查询失败，服务器异常！", null);
            //3.2：使用工具类向前端响应json数据
            JsonUtils.printResult(response,result);
        }finally {
            JedisUtils.close(jedis);
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doPost(request, response);
    }
}
```

- ProvinceService

```
public class ProvinceService {
    public List<Province> findAll() throws SQLException {
        ProvinceDao provinceDao = new ProvinceDao();
        return provinceDao.findAll();
    }
}
```

- ProvinceDao

```
public class ProvinceDao {
    public List<Province> findAll() throws SQLException {
        //调用DBUtils操作数据库
        QueryRunner queryRunner = new QueryRunner(C3P0Utils.getDataSource());
        String sql ="select * from province";
        return queryRunner.query(sql, new BeanListHandler<Province>
(Province.class));
    }
}
```

4.小结

1. 优化
    - 先判断redis中是否存储数据
    - 存在：直接从redis中获取　注意：redis中存的String类型 所以取出后需要转换一下

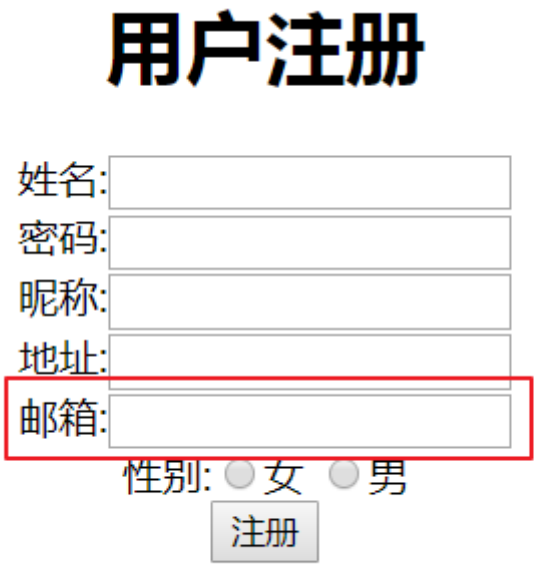
      ```
      //注意2：存入redis的是json字符串 存进去之后又变成了普通String　需要转回来，不转的话前端无法正常解析
      Result result = new Result(true,"从redis中查询成功！",
      JSON.parse(province_list));
      ```

    - 不存在：从MySQL中查询，查询之后存入redis，然后响应

```
//注意1:在将数据存入redis中时  需要转成json字符串存入
jedis.set("province_list",JSON.toJSONString(list));
```

案例-使用Redis完成邮箱验证码的校验

1.需求



- 输入邮箱, 发送激活邮件, 完成校验

2.分析

2.1发送邮件思路

1. 填写邮箱地址，发送激活链接(附加激活码)
2. 将生成的激活码存入redis，24h
3. 发送激活邮件到用户邮箱

2.2验证思路

1. 创建一个激活Servlet activeServlet
2. 获取redis中的验证码与激活链接的验证码进行比对
3. 激活成功(更新账号状态status为1，表示启用 删除redis中存储的该验证码)

3.代码实现

3.1发送邮件代码

- RegisterServlet添加发送邮件代码

```
/*******************************注册时发送激活邮件
****************************************/
 //2.1：生成激活码
String activeCode = UUIDUtils.generateUUID();
```

```
//2.2：将激活码保存到redis key使用email
Jedis jedis = JedisUtils.getJedis();
JedisUtils.close(jedis);
jedis.setex(user.getEmail(),60*60*24,activeCode);
//2.3：发送激活邮件【拼接激活链接地址】
http://localhost:8080/active?
key=ls@itheima.com&code=a895d7313d654ab080b1d492a5207d1d
MailUtil.sendMail(user.getEmail(),
  "账号激活","尊敬的用户，您好，感谢你注册成为本网站用户，" +
  "请通过激活链接<ahref='http://localhost:8080"+request.getContextPath()+"/active?
key="+user.getEmail()+"&code="+activeCode+"'>激活</a>，启用您的账号。");

/*****************************注册时发送激活邮件
****************************************/
```

3.2验证代码

- ActiveServlet

```
package com.itheima.web;

import com.itheima.utils.JedisUtils;
import redis.clients.jedis.Jedis;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(value = "/active")
public class ActiveServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        //2.1：从激活链接中获取激活码
        String email = request.getParameter("key");
        String code = request.getParameter("code");
        //2.2：从redis中获取保存的激活码
        Jedis jedis = JedisUtils.getJedis();
        Boolean flag = jedis.exists(email);
        if(flag){
            String activeCode = jedis.get(email);
            //2.3：判断 比对是否一致
            if(activeCode.equals(code)){
                //2.4：一致 激活成功(①修改激活状态status ②删除redis中的激活码 ③跳转到登录
页面)
                response.getWriter().print("<script>alert('激活成
功！');location.href='login.html';</script>");
            }else{
                //2.5：激活失败
```

```
                response.getWriter().print("激活失败！");
            }
        }else{
            //2.5：激活失败
            response.getWriter().print("激活失败！");
        }

    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doPost(request, response);
    }
}
```

4.小结

- 注册时 发送激活链接邮件到邮箱 ==注意==：要将激活码存入redis
- 编写激活Servlet 在其中获取激活链接传递的激活码和redis中存储的激活码进行比对，一致就说明激活成功！
- **练习：下去之后 自己完成激活成功后 1.更新账号状态status=1 2.删除对应的redis key**