# 1.redis安装与配置

## 1.1 redis简介

redis是远程数据库，分为客户端和服务端，通常所说的redis是其服务端。

redis是基于内存的数据库，其所有的数据都存储在内存中。

redis是一个非关系型数据库，它不需要定义数据字典(定义表)就可以使用

## 1.2 redis的应用

redis由于存储在内存中，因此操作速度非常快，因此我们常把它作为缓存使用

redis可以作为发布-订阅式的消息队列使用

我们可以直接把redis用作数据存储，也即是可以对其进行增删改查

## 1.3 redis的安装和使用

redis的安装环境是linux下

参考<http://www.cnblogs.com/hanyinglong/p/5036558.html>

下载redis <http://download.redis.io/releases/redis-3.0.5.tar.gz>

下载tcl <http://downloads.sourceforge.net/tcl/tcl8.6.1-src.tar.gz>

安装完成后，在安装路径下可执行的相关操作如下：

**src/redis-server &** 启动redis，加上&表示使redis以后台程序方式运行

**redis-server /usr/local/kencery/redis/redis.conf** 启动redis

**src/redis-cli shutdown**  关闭redis

**systemctl enable redis.service** 设置redis开机自启

下面两个指令是linux下直接执行，不需要进入安装路径，但需要配置相关文件，配置方法参见<http://www.cnblogs.com/hanyinglong/p/5036558.html>第七节

**service redis start 开启redis服务**

**service redis stop 关闭redis服务**

如果要使用redis，请输入**redis-cli**指令连接redis数据库

通过rpm -ql redis查找redis.conf位置，打开/redis.conf 里面注释掉这个： #bind 127.0.0.1，并搜索requirepass，将requirepass后面的参数修改为redis密码，最后别忘了重启reids

# 2.redis数据类型

## 2.1 redis的数据类型

redis支持五种数据类型：string（字符串），hash（哈希），list（列表），set（集合）及sort set (有序集合)



## 2.2 字符串string

string是redis最基本的类型，是一种二进制安全的(可以包含任何数据。比如jpg图片或者序列化的对象), 一个key对应一个value。



**语法：**

**set keyName value //给keyName赋值value**

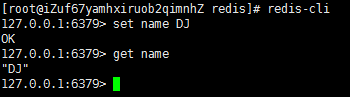
**get keyName //获取keyName的值**

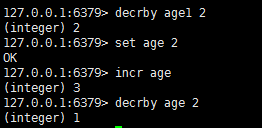
**strlen keyName //获取keyName的值的长度**

**incr keyName //如果keyName是整数，则自增1，否则报错**

**decry keyName num //值为keyName-num**

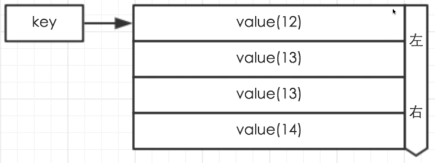
[**http://www.runoob.com/redis/redis-strings.html**](http://www.runoob.com/redis/redis-strings.html)





## 2.3 列表list

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）



**语法：**

**LPUSH list value //从list左(上)边推入一个value**

**RPUSH list value //从list右(下)边推入一个value**

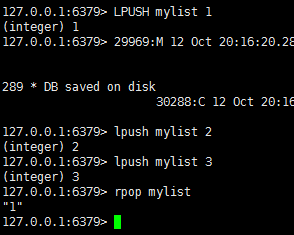
**LPOP list value //从list左(上)边取出一个value**

**RPOP list value //从list右(下)边取出一个value**

**llen list //获取list的长度**

[**http://www.runoob.com/redis/redis-lists.html**](http://www.runoob.com/redis/redis-lists.html)

**eg:**

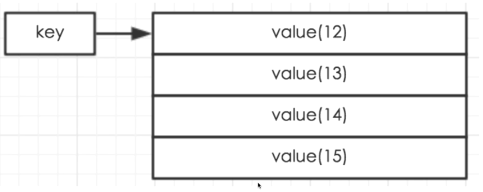


**注意： 1.list和set不同，它不要求value是唯一的(不重复),并且list是有序存储，而set是无序存储**

**2.pop操作后，元素就不在list中了**

## 2.4 集合set

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。由于set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)



**语法：**

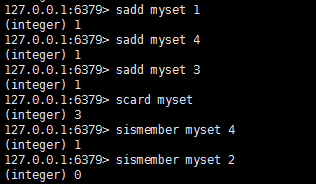
**SADD set value //往set中插入value，插入重复元素是不会生效的**

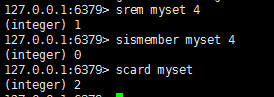
**srem set value //从set中删除value**

**smembers set //返回set中所有元素**

**sismember set value //查看set中是否存在value元素，存在返回1，不存在返回0**

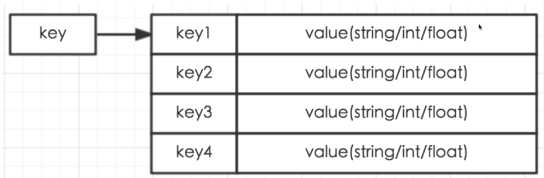
**SCARD set //查看set中有几个元素**





## 2.5 哈希hash

Redis hash 是一个key和value的映射表，它的键值key是string类型，hash特别适合用于存储对象。



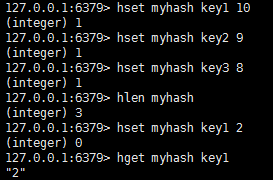
语法：

**hset hash key value //向hash中插入key-value键值对**

**hget hash key //从hash中取出key的值**

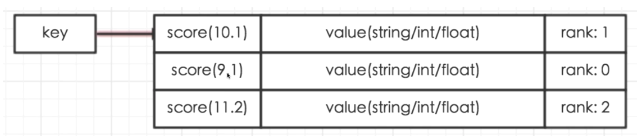
**hlen hash //查看hash中有几个元素**

[**http://www.runoob.com/redis/redis-hashes.html**](http://www.runoob.com/redis/redis-hashes.html)



## 2.6 有序集合zset

zset和set很像，但它的每个元素多了一个分数用作排序,redis会按照这个分数(键值)自动给zset中的元素进行排序，和set相同的是**zset的value必须是唯一的**，不能重复，**如果出现了分数(score)相同的情形，rank排序按照value的字典顺序进行，排名从0开始**



**语法：**

**ZADD zset score value //往zset中插入score-value对，value不能重复，score可以重复(重复后按value字典序排序)**

**zrem zset value //从zset中删除value**

**zrank zset value //查看value的排名**

**zrange zset 0 2 //从zset中获取排名0-2的元素**

**ZCARD zset //查看zset中有几个元素**

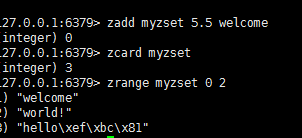
eg：zadd myzset 9.1 hello！

zadd myzset 8.4 world!

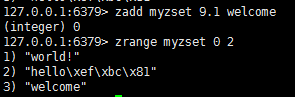
zadd myzset 12.3 welcome

zrange myzset 0 2 //输出world! hello! welcome

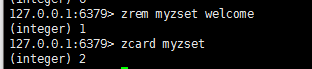
**修改value的键值**



**按字典顺序排列**



**删除元素**



# 3.PHP中的redis

## 3.1 php redis扩展环境安装

安装解压工具unzip：**yum install -y unzip zip**

从<https://pan.baidu.com/s/1i5DFrjn?errno=0&errmsg=Auth%20Login%20Sucess&&bduss=&ssnerror=0>获取php7-redis扩展包

**unzip phpredis-php7.zip**解压

cd进入解压后的文件夹,输入 **phpize**，用**phpize**生成configure配置文件

输入**./configure**，完成后输入**make**进行编译，最后输入**make install**安装

输入**vim php安装地址/etc/php.in**进行配置，在最后一行（G直接到最后一行）添加以下内容extension="redis.so"

## 3.2 php链接redis

php提供了redis类来对redis进行操作，redis类的命名空间是根目录，**ps auxf | grep redis-server**指令可以查看redis服务端的端口号

redis类的**connect(redis的ip,端口号)**方法可以链接redis数据库

eg:

$redis=new \Redis(); //新建Redis类，\代表命名空间为根目录

$redis->connect('127.0.0.1', 6379); //链接数据库

/\*1.Connection\*/

$redis = new Redis();

$redis->connect('127.0.0.1',6379,1);//短链接，本地host，端口为6379，超过1秒放弃链接

$redis->open('127.0.0.1',6379,1);//短链接(同上)

$redis->pconnect('127.0.0.1',6379,1);//长链接，本地host，端口为6379，超过1秒放弃链接

$redis->popen('127.0.0.1',6379,1);//长链接(同上)

$redis->auth('password');//登录验证密码，返回【true | false】

$redis->select(0);//选择redis库,0~15 共16个库

$redis->close();//释放资源

$redis->ping(); //检查是否还再链接,[+pong]

$redis->ttl('key');//查看失效时间[-1 | timestamps]

$redis->persist('key');//移除失效时间[ 1 | 0]

$redis->sort('key',[$array]);//返回或保存给定列表、集合、有序集合key中经过排序的元素，$array为参数limit等！【配合$array很强大】 [array|false]

## 3.3 string相关方法

//string

$redis->setOption(Redis::OPT\_PREFIX,'hf\_');//设置表前缀为hf\_

$redis->set('key',1);//设置key=aa value=1 [true]

$redis->mset($arr);//设置一个或多个键值[true]

$redis->setnx('key','value');//key=value,key存在返回false[|true]

$redis->get('key');//获取key [value]

$redis->mget($arr);//(string|arr),返回所查询键的值

$redis->del($key\_arr);//(string|arr)删除key，支持数组批量删除【返回删除个数】

$redis->delete($key\_str,$key2,$key3);//删除keys,[del\_num]

$redis->getset('old\_key','new\_value');//先获得key的值，然后重新赋值,[old\_value | false]

$redis->strlen('key');//获取当前key存储的值的长度

$redis->append('key','string');//把string的值追加到key现有的value中[追加后的个数]

$redis->incr('key');//自增1，如不存在key,赋值为1(只对整数有效,存储以10进制64位，redis中为str)[new\_num | false]

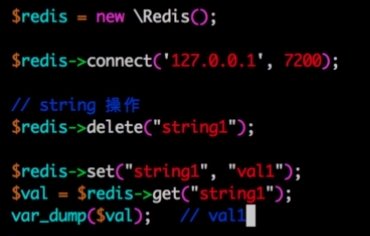
$redis->incrby('key',$num);//自增$num,不存在为赋值,值需为整数[new\_num | false]

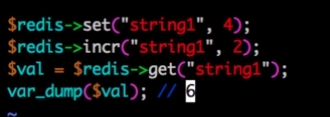
$redis->decr('key');//自减1，[new\_num | false]

$redis->decrby('key',$num);//自减$num，[ new\_num | false]

$redis->setex('key',10,'value');//key=value，有效期为10秒[true]

eg;





## 3.4 list相关方法

//list

$redis->llen('key');//返回列表key的长度,不存在key返回0， [ len | 0]

$redis->lpush('key','value');//增，只能将一个值value插入到列表key的表头，不存在就创建 [列表的长度 |false]

$redis->rpush('key','value');//增，只能将一个值value插入到列表key的表尾 [列表的长度 |false]

$redis->lInsert('key', Redis::AFTER, 'value', 'new\_value');//增，将值value插入到列表key当中，位于值value之前或之后。[new\_len | false]

$redis->lpushx('key','value');//增，只能将一个值value插入到列表key的表头，不存在不创建 [列表的长度 |false]

$redis->rpushx('key','value');//增，只能将一个值value插入到列表key的表尾，不存在不创建 [列表的长度 |false]

$redis->lpop('key');//删，移除并返回列表key的头元素,[被删元素 | false]

$redis->rpop('key');//删，移除并返回列表key的尾元素,[被删元素 | false]

$redis->lrem('key','value',0);//删，根据参数count的值，移除列表中与参数value相等的元素count=(0|-n表头向尾|+n表尾向头移除n个value) [被移除的数量 | 0]

$redis->ltrim('key',start,end);//删，列表修剪，保留(start,end)之间的值 [true|false]

$redis->lset('key',index,'new\_v');//改，从表头数，将列表key下标为第index的元素的值为new\_v, [true | false]

$redis->lindex('key',index);//查，返回列表key中，下标为index的元素[value|false]

$redis->lrange('key',0,-1);//查，(start,stop|0,-1)返回列表key中指定区间内的元素，区间以偏移量start和stop指定。[array|false]

## 3.5 set相关方法

//set

$redis->scard('key');//返回集合key的基数(集合中元素的数量)。[num | 0]

$redis->sMove('key1', 'key2', 'member');//移动，将member元素从key1集合移动到key2集合。[1 | 0]

$redis->sadd('key','value1','value2','valuen');//增，改，将一个或多个member元素加入到集合key当中，已经存在于集合的member元素将被忽略。[insert\_num]

$redis->srem('key','value1','value2','valuen');//删，移除集合key中的一个或多个member元素，不存在的member元素会被忽略 [del\_num | false]

$redis->smembers('key');//查，返回集合key中的所有成员 [array | '']

$redis->sismember('key','member');//判断member元素是否是集合key的成员 [1 | 0]

$redis->spop('key');//删，移除并返回集合中的一个随机元素 [member | false]

$redis->srandmember('key');//查，返回集合中的一个随机元素 [member | false]

$redis->sinter('key1','key2','keyn');//查，返回所有给定集合的交集 [array | false]

$redis->sunion('key1','key2','keyn');//查，返回所有给定集合的并集 [array | false]

$redis->sdiff('key1','key2','keyn');//查，返回所有给定集合的差集 [array | false]

## 3.6 hash相关方法

//hash

$redis->hexists('key','field');//查看hash中是否存在field,[1 | 0]

$redis->hincrby('key','field',$int\_num);//为哈希表key中的域field的值加上量(+|-)num,[new\_num | false]

$redis->hlen('key');//返回哈希表key中域的数量。[ num | 0]

$redis->hset('key','field','value');//增，改，将哈希表key中的域field的值设为value,不存在创建,存在就覆盖【1 | 0】

$redis->hget('key','field');//查，取值【value|false】

$arr = array('one'=>1,2,3);$arr2 = array('one',0,1);

$redis->hmset('key',$arr);//增，改，设置多值$arr为(索引|关联)数组,$arr[key]=field, [ true ]

$redis->hmget('key',$arr2);//查，获取指定下标的field，[$arr | false]

$redis->hgetall('key');//查，返回哈希表key中的所有域和值。[当key不存在时，返回一个空表]

$redis->hkeys('key');//查，返回哈希表key中的所有域。[当key不存在时，返回一个空表]

$redis->hvals('key');//查，返回哈希表key中的所有值。[当key不存在时，返回一个空表]

$redis->hdel('key',$arr2);//删，删除指定下标的field,不存在的域将被忽略,[num | false]

## 3.7 zset相关方法

//Zset

$redis->zcard('key');//返回集合key的基数(集合中元素的数量)。[num | 0]

$redis->zcount('key',0,-1);//返回有序集key中，score值在min和max之间(默认包括score值等于min或max)的成员。[num | 0]

$redis->zAdd('key',$score1,$member1,$scoreN,$memberN);//增，改，将一个或多个member元素及其score值加入到有序集key当中。[num | 0]

$redis->zrem('key','member1','membern');//删，移除有序集key中的一个或多个成员，不存在的成员将被忽略。[del\_num | 0]

$redis->zscore('key','member');//查,通过值反拿权 [num | null]

$redis->zrange('key',$start,$stop);//查，通过(score从小到大)【排序名次范围】拿member值，返回有序集key中，【指定区间内】的成员 [array | null]

$redis->zrevrange('key',$start,$stop);//查，通过(score从大到小)【排序名次范围】拿member值，返回有序集key中，【指定区间内】的成员 [array | null]

$redis->zrangebyscore('key',$min,$max[,$config]);//查，通过scroe权范围拿member值，返回有序集key中，指定区间内的(从小到大排)成员[array | null]

$redis->zrevrangebyscore('key',$max,$min[,$config]);//查，通过scroe权范围拿member值，返回有序集key中，指定区间内的(从大到小排)成员[array | null]

$redis->zrank('key','member');//查，通过member值查(score从小到大)排名结果中的【member排序名次】[order | null]

$redis->zrevrank('key','member');//查，通过member值查(score从大到小)排名结果中的【member排序名次】[order | null]

$redis->ZINTERSTORE();//交集

$redis->ZUNIONSTORE();//差集

## 3.8 server相关方法

/\*3.Server\*/

$redis->dbSize();//返回当前库中的key的个数

$redis->flushAll();//清空整个redis[总true]

$redis->flushDB();//清空当前redis库[总true]

$redis->save();//同步??把数据存储到磁盘-dump.rdb[true]

$redis->bgsave();//异步？？把数据存储到磁盘-dump.rdb[true]

$redis->info();//查询当前redis的状态 [verson:2.4.5....]

$redis->lastSave();//上次存储时间key的时间[timestamp]

$redis->watch('key','keyn');//监视一个(或多个) key ，如果在事务执行之前这个(或这些) key 被其他命令所改动，那么事务将被打断 [true]

$redis->unwatch('key','keyn');//取消监视一个(或多个) key [true]

$redis->multi(Redis::MULTI);//开启事务，事务块内的多条命令会按照先后顺序被放进一个队列当中，最后由 EXEC 命令在一个原子时间内执行。

$redis->multi(Redis::PIPELINE);//开启管道，事务块内的多条命令会按照先后顺序被放进一个队列当中，最后由 EXEC 命令在一个原子时间内执行。

$redis->exec();//执行所有事务块内的命令，；【事务块内所有命令的返回值，按命令执行的先后顺序排列，当操作被打断时，返回空值 false】

# 4.java使用redis

## 4.1驱动包jedis

如果想要在java中使用redis，应当导入jedis包，其maven代码为：

<dependency>  
 <groupId>redis.clients</groupId>  
 <artifactId>jedis</artifactId>  
 <version>${jedisVersion}</version>  
</dependency>

如果需要与spring整合，则还需要添加对应的jar包

<dependency>  
 <groupId>org.springframework.data</groupId>  
 <artifactId>spring-data-redis</artifactId>  
 <version>${springRedis}</version>  
</dependency>

## 4.2使用jedis操作redis

### 4.2.1新建连接

Jedis对象的第一个参数是redis服务器ip，第二个参数可选，是redis的端口号，默认为6379

使用完毕别忘记关闭连接

Jedis jedis = new Jedis("101.132.43.150",6379);  
jedis.auth("2571838183");  
System.*out*.println("连接成功");  
//查看服务是否运行  
System.*out*.println("服务正在运行: "+jedis.ping());

jedis.quit();

### 4.2.2使用jedis存储string

set(key, value)：给数据库中名称为key的string赋予值value  
    get(key)：返回数据库中名称为key的string的value  
    getset(key, value)：给名称为key的string赋予上一次的value  
    mget(key1, key2,…, key N)：返回库中多个string的value  
    setnx(key, value)：添加string，名称为key，值为value  
    setex(key, time, value)：向库中添加string，设定过期时间time  
    mset(key N, value N)：批量设置多个string的值  
    msetnx(key N, value N)：如果所有名称为key i的string都不存在  
    incr(key)：名称为key的string增1操作  
    incrby(key, integer)：名称为key的string增加integer  
    decr(key)：名称为key的string减1操作  
    decrby(key, integer)：名称为key的string减少integer  
    append(key, value)：名称为key的string的值附加value  
    substr(key, start, end)：返回名称为key的string的value的子串

//存储字符串  
jedis.set("redisString","这是存储在redis中的字符串");  
//输出string类型，utf8中汉字3个字节，因此长度是35  
System.*out*.println("字符串: "+jedis.get("redisString")+"。字符串长度为："+jedis.strlen("redisString"));

### 4.2.3使用jedis存储list

rpush(key, value)：在名称为key的list尾添加一个值为value的元素  
    lpush(key, value)：在名称为key的list头添加一个值为value的 元素  
    llen(key)：返回名称为key的list的长度  
    lrange(key, start, end)：返回名称为key的list中start至end之间的元素  
    ltrim(key, start, end)：截取名称为key的list  
    lindex(key, index)：返回名称为key的list中index位置的元素  
    lset(key, index, value)：给名称为key的list中index位置的元素赋值  
    lrem(key, count, value)：删除count个key的list中值为value的元素  
    lpop(key)：返回并删除名称为key的list中的首元素  
    rpop(key)：返回并删除名称为key的list中的尾元素  
    blpop(key1, key2,… key N, timeout)：lpop命令的block版本。  
    brpop(key1, key2,… key N, timeout)：rpop的block版本。  
    rpoplpush(srckey, dstkey)：返回并删除名称为srckey的list的尾元素，

//存储list  
jedis.del("redisList");  
jedis.lpush("redisList","界");  
jedis.lpush("redisList","世");  
jedis.lpush("redisList","好");  
jedis.lpush("redisList","你");  
  
//输出list类型  
int listLength=jedis.llen("redisList").intValue();  
List<String> list=new ArrayList<String>();  
for (int i=0;i<listLength;i++){  
 list.add(i,jedis.lpop("redisList"));  
}  
System.*out*.println(list);

### 4.2.4使用jedis存储set

sadd(key, member)：向名称为key的set中添加元素member  
    srem(key, member) ：删除名称为key的set中的元素member  
    spop(key) ：随机返回并删除名称为key的set中一个元素  
    smove(srckey, dstkey, member) ：移到集合元素  
    scard(key) ：返回名称为key的set的基数  
    sismember(key, member) ：member是否是名称为key的set的元素  
    sinter(key1, key2,…key N) ：求交集  
    sinterstore(dstkey, (keys)) ：求交集并将交集保存到dstkey的集合  
    sunion(key1, (keys)) ：求并集  
    sunionstore(dstkey, (keys)) ：求并集并将并集保存到dstkey的集合  
    sdiff(key1, (keys)) ：求差集  
    sdiffstore(dstkey, (keys)) ：求差集并将差集保存到dstkey的集合  
    smembers(key) ：返回名称为key的set的所有元素  
    srandmember(key) ：随机返回名称为key的set的一个元素

//存储set类型  
jedis.del("redisSet");  
jedis.sadd("redisSet","你好");  
jedis.sadd("redisSet","世界");  
jedis.sadd("redisSet","你好");  
Set redisSet=jedis.smembers("redisSet");  
int size=jedis.scard("redisSet").intValue();  
System.*out*.println("一共"+size+"个元素"+redisSet);  
jedis.sadd("redisSet","!");  
jedis.srem("redisSet","世界");  
redisSet=jedis.smembers("redisSet");  
System.*out*.println(redisSet);

### 4.2.5使用jedis存储hash

   hset(key, field, value)：向名称为key的hash中添加元素field  
    hget(key, field)：返回名称为key的hash中field对应的value  
    hmget(key, (fields))：返回名称为key的hash中field i对应的value  
    hmset(key, (fields))：向名称为key的hash中添加元素field   
    hincrby(key, field, integer)：将名称为key的hash中field的value增加integer  
    hexists(key, field)：名称为key的hash中是否存在键为field的域  
    hdel(key, field)：删除名称为key的hash中键为field的域  
    hlen(key)：返回名称为key的hash中元素个数  
    hkeys(key)：返回名称为key的hash中所有键  
    hvals(key)：返回名称为key的hash中所有键对应的value  
    hgetall(key)：返回名称为key的hash中所有的键（field）及其对应的value

//存储hash类型  
jedis.hset("redisHash","key1","value1");  
jedis.hset("redisHash","key2","value2");  
jedis.hset("redisHash","key3","value3");  
jedis.hdel("redisHash","key2");  
Map redisHash=jedis.hgetAll("redisHash");  
System.*out*.println(redisHash);

### 4.2.5使用jedis存储zset

//连接本地的 Redis 服务  
Jedis jedis = new Jedis("101.132.43.150",6379);  
jedis.auth("2571838183");  
jedis.zadd("redisZset",5.0,"value1");  
jedis.zadd("redisZset",5,"value2");  
jedis.zadd("redisZset",4.0,"value3");  
jedis.zadd("redisZset",4.5,"value4");  
int size=jedis.zcard("redisZset").intValue();  
int value4Rank=jedis.zrank("redisZset","value4").intValue();  
System.*out*.println("长度:"+size+"，value4的排名为："+value4Rank);

## 4.3spring集成redis

首先我们需要引入相关jar包

<dependency>  
 <groupId>org.springframework.data</groupId>  
 <artifactId>spring-data-redis</artifactId>  
 <version>${spring.redis.version}</version>  
</dependency>  
<dependency>  
 <groupId>redis.clients</groupId>  
 <artifactId>jedis</artifactId>  
 <version>${jedis.version}</version>  
</dependency>

配置redis属性

#访问地址  
redis.host=101.132.43.150  
#访问端口  
redis.port=6379  
#注意，如果没有password，此处不设置值，但这一项要保留  
redis.password=2571838183  
  
#最大空闲数，数据库连接的最大空闲时间。超过空闲时间，数据库连接将被标记为不可用，然后被释放。设为0表示无限制。  
redis.maxIdle=300  
#连接池的最大数据库连接数。设为0表示无限制  
redis.maxActive=600  
#最大建立连接等待时间。如果超过此时间将接到异常。设为-1表示无限制。  
redis.maxWait=1000  
#在borrow一个jedis实例时，是否提前进行alidate操作；如果为true，则得到的jedis实例均是可用的；  
redis.testOnBorrow=true

Spring配置文件

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:p="http://www.springframework.org/schema/p"  
 xmlns:tx="http://www.springframework.org/schema/tx"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd  
 http://www.springframework.org/schema/tx  
 http://www.springframework.org/schema/tx/spring-tx-3.0.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context-3.0.xsd">  
  
 <!-- scanner redis properties -->  
 <context:property-placeholder location="classpath:properties/redis.properties" ignore-unresolvable="true"/>  
  
 <!--（1）如果你有多个数据源需要通过<context:property-placeholder管理，且不愿意放在一个配置文件里，那么一定要加上ignore-unresolvable=“true"-->  
 <!--（2）注意新版的（具体从哪个版本开始不清楚，有兴趣可以查一下）JedisPoolConfig的property name，不是maxActive而是maxTotal，而且没有maxWait属性，建议看一下Jedis源码。-->  
 <!-- redis连接池 -->  
 <bean id="jedisConfig" class="redis.clients.jedis.JedisPoolConfig">  
 <property name="maxTotal" value="${redis.maxActive}"></property>  
 <property name="maxIdle" value="${redis.maxIdle}"></property>  
 <property name="maxWaitMillis" value="${redis.maxWait}"></property>  
 <property name="testOnBorrow" value="${redis.testOnBorrow}"></property>  
 </bean>  
 <!-- redis连接工厂 -->  
 <bean id="connectionFactory" class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory">  
 <property name="hostName" value="${redis.host}"></property>  
 <property name="port" value="${redis.port}"></property>  
 <property name="password" value="${redis.password}"></property>  
 <property name="poolConfig" ref="jedisConfig"></property>  
 </bean>  
 <!-- redis操作模板，这里采用尽量面向对象的模板 -->  
 <bean id="redisTemplate" class="org.springframework.data.redis.core.StringRedisTemplate">  
 <property name="connectionFactory" ref="connectionFactory"/>  
 <!-- 如果不配置Serializer，那么存储的时候只能使用String，如果用对象类型存储，那么会提示错误 can't cast to String！！！-->  
 <property name="keySerializer">  
 <bean class="org.springframework.data.redis.serializer.StringRedisSerializer"/>  
 </property>  
 <property name="valueSerializer">  
 <bean class="org.springframework.data.redis.serializer.JdkSerializationRedisSerializer"/>  
 </property>  
 <!--开启事务-->  
 <property name="enableTransactionSupport" value="true"/>  
 </bean>  
</beans>

# 5.其他问题

## 5.1 redis的缓存方案