# docker mysql redis

## 一、获取 redis 镜像

docker search redis

docker pull redis:5.0.5

docker images

## 二、创建容器

创建宿主机 redis 容器的数据和配置文件目录

# 这里我们在 /home/docker 下创建

mkdir /home/docker/redis/{conf,data} -p

cd /home/docker/redis

**注意：后面所有的操作命令都要在这个目录/home/docker/redis下进行**

获取 redis 的默认配置模版

# 获取 redis 的默认配置模版

# 这里主要是想设置下 redis 的 log / password / appendonly

# redis 的 docker 运行参数提供了 --appendonly yes 但没 password

wget https://raw.githubusercontent.com/antirez/redis/4.0/redis.conf -O conf/redis.conf

# 直接替换编辑

sed -i 's/logfile ""/logfile "access.log"/' conf/redis.conf;

sed -i 's/# requirepass foobared/requirepass 123456/' conf/redis.conf;

sed -i 's/appendonly no/appendonly yes/' conf/redis.conf;

sed -i 's/bind 127.0.0.1/bind 0.0.0.0/' conf/redis.conf;

protected-mode 是在没有显式定义 bind 地址（即监听全网段），又没有设置密码 requirepass时，protected-mode 只允许本地回环 127.0.0.1 访问。改为bind 0.0.0.0

创建并运行一个名为 myredis 的容器,放到start-redis.sh脚本里面

# 创建并运行一个名为 myredis 的容器

docker run \

-p 6379:6379 \

-v $PWD/data:/data \

-v $PWD/conf/redis.conf:/etc/redis/redis.conf \

--privileged=true \

--name myredis \

-d redis:5.0.5 redis-server /etc/redis/redis.conf

# 命令分解

docker run \

-p 6379:6379 \ # 端口映射 宿主机:容器

-v $PWD/data:/data:rw \ # 映射数据目录 rw 为读写

-v $PWD/conf/redis.conf:/etc/redis/redis.conf:ro \ # 挂载配置文件 ro 为readonly

--privileged=true \ # 给与一些权限

--name myredis \ # 给容器起个名字

-d redis redis-server /etc/redis/redis.conf # deamon 运行容器 并使用配置文件启动容器内的 redis-server

查看活跃的容器

# 查看活跃的容器

docker ps

# 如果没有 myredis 说明启动失败 查看错误日志

docker logs myredis

# 查看 myredis 的 ip 挂载 端口映射等信息

docker inspect myredis

# 查看 myredis 的端口映射

docker port myredis

## 三、访问 redis 容器服务

docker exec -it myredis bash

redis-cli

## 四、开启防火墙端口，提供外部访问

firewall-cmd --zone=public --add-port=6379/tcp --permanent

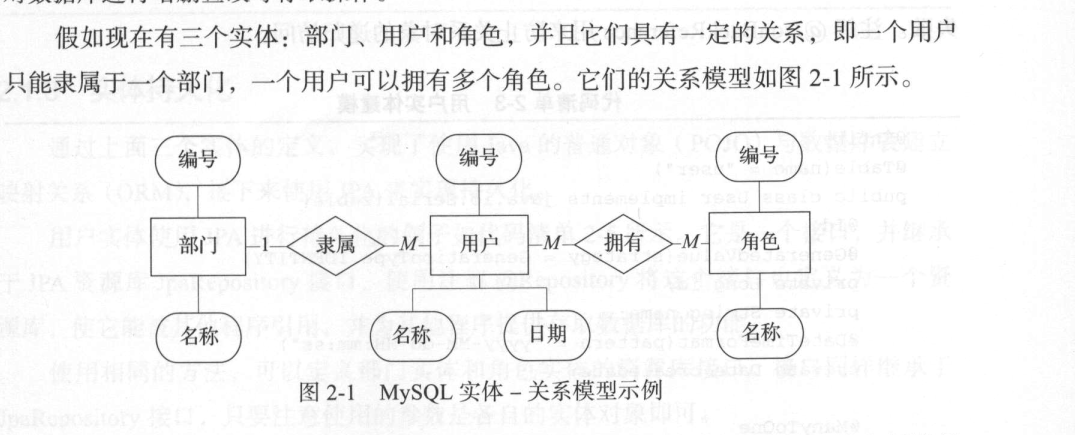
firewall-cmd --reload

firewall-cmd --query-port=6379/tcp

# mysql

## 依赖

## 数据库



## 对象序列化的用途

1、Java程序在运行中，对象都是分配在内存中，而序列化的一个用途就是将内存的中对象转化成磁盘中的对象。

2、对象不能直接在网络中传输，有时候需要将对象转成字节序列在网络中传输，这就是对象序列化的第二个用途。

manytomany

fetch = FetchType.EAGER 作用

如果是EAGER，那么表示取出这条数据时，它关联的数据也同时取出放入内存

如果是LAZY，那么取出这条数据时，它关联的数据并不取出来，在同一个session中，什么时候要用，就什么时候取(再次访问数据库)

但是，在session外，就不能再取了。用EAGER时，因为在内存里，所以在session外也可以取。

cascade

<https://www.cnblogs.com/hdwang/p/7241043.html>

必须级联到关联目标的操作。默认情况下没有级联操作。

这里解释一下，级联的意思是：本实体做了什么事，也要拉上 另一个关联的实体，导致另一个实体跟着做事情。就是说我删除了，你也得删除！ 关联目标，指的是 关联的那个实体。

在有中间关系表的情况下，比如 user/role/user\_role 这三张表，只对应User/Role两个实体对象就够了！中间表是由User对象或者Role对象维护的，不用另外新建user\_role的实体！

不级联就是说，比如在User类中，@ManyToMany没有写cascade就是说，User的操作不影响Role的操作，也就是说针对user表的操作不影响role表的记录。这个和user\_role关系表没关系！user\_role表的映射关系的维护，和cascade写不写，写什么值一点关系都没有！hibernate会给你维护好关系表的记录的,经测试是先删除关系记录，再插入关系记录。

今天学习公司代码发现@Order(Ordered.HIGHEST\_PRECEDENCE)这个注解  
了解了一下  
我的pom4.0.0说明是spring4  
Spring 4.0中默认情况下遵循从低到高的顺序，即最低值具有高优先级。 这意味着它们首先出现在列表或数组中。 因为默认情况下，排序优先级为LOWEST\_PRECEDENCE。 如果您首先需要最高值，那么我们需要将此值更改为Ordered.HIGHEST\_PRECEDENCE。

sort

<https://blog.csdn.net/qq_44039966/article/details/102713779?depth_1-utm_source=distribute.pc_relevant.none-task&utm_source=distribute.pc_relevant.none-task>



GSON弥补了JSON的许多不足的地方，在实际应用中更加适用于Java开发。在这里，我们主要讲解的是利用GSON来操作java对象和json数据之间的相互转换，包括了常见的对象序列化和反序列化的知识。

GSON

## 一、前言

因为json有2种类型：

* 一种是对象，object -> {key:value,key:value,...} 。
* 另一种是数组，array -> [value,value,...] 。

所以针对这两种类型，来展开对json数据的操作。

GSON在解析json的时候，大体上有2种类型，一种是直接在内存中生成object或array，通过手工指定key来获取值；另一种是借助javabean来进行映射获取值。

## 二、对 json 数据进行反序列化，得到java 对象

### 1、不借助java 类，直接解析json 数据

#### 1、json 是对象类型

当ajax传过来的json数据属于对象时，不论这个对象简单还是复杂，都可以轻松地把它们给解析出来。

ajax传过来的json数据（是对象形式）：

作者：7叶  
链接：https://www.jianshu.com/p/75a50aa0cad1  
来源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

# redis

[https://docs.spring.io/spring-data/data-redis/docs/current/reference/html/#](https://docs.spring.io/spring-data/data-redis/docs/current/reference/html/)

<https://zhuanlan.zhihu.com/p/68059953>

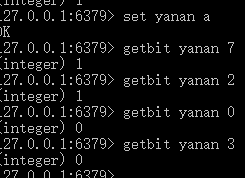
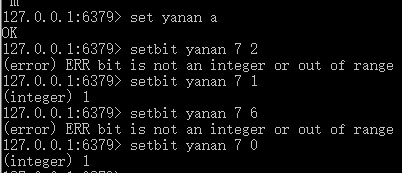
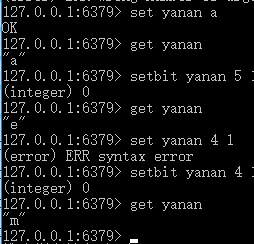
[**redis五大类型用法**](https://www.cnblogs.com/yanan7890/p/6617305.html)

Redis五大类型:字符串（String）、哈希/散列/字典（Hash）、列表（List）、集合（Set）、有序集合（sorted set）五种  
Controller:@Resource RedisTemplate<String, String> redisTemplate;  
总括:  
redisTemplate.opsForValue();//操作字符串  
redisTemplate.opsForHash();//操作hash  
redisTemplate.opsForList();//操作list  
redisTemplate.opsForSet();//操作set  
redisTemplate.opsForZSet();//操作有序set  
  
String:

1. redisTemplate.opsForValue().set(key,value));   
   2.redisTemplate.opsForValue().get(key));   
   3.redisTemplate.opsForValue().get(key, start, end);  
   4.redisTemplate.opsForValue().getAndSet(key, value);  
   5.redisTemplate.opsForValue().getBit(key, offset);//下方注释  
   6.redisTemplate.opsForValue().multiGet(keys);  
   7.redisTemplate.opsForValue().setBit(key, offset, value);//下方注释  
   8.redisTemplate.opsForValue().set(K key, V value, long timeout, TimeUnit unit);//TimeUnit是timeout的类型,如毫秒\秒\天等  
   9.redisTemplate.opsForValue().setIfAbsent(key, value);  
   10.redisTemplate.opsForValue().set(K key, V value, long offset);//博主此处未做java验证  
   11.redisTemplate.opsForValue().size(key));  
   12.redisTemplate.opsForValue().multiGet(Collection<K> keys);  
   13.redisTemplate.opsForValue().multiSetIfAbsent(Map<? extends K, ? extends V> m);  
   14.同8  
   15\16\17\18\19.redisTemplate.opsForValue().increment(K key, long delta);或.increment(K key, double delta);  
   20.redisTemplate.opsForValue().append(key, value);//在key键对应值的右面追加值value  
   可以看到并没有删除等方法,博主研究了一下可以这样:21.del key------21.redisTemplate.opsForValue().getOperations().delete(key);

mongoDB

| **编号** | **命令** | **描述说明** |
| --- | --- | --- |
| 1 | [SET key value](http://www.yiibai.com/redis/strings_set.html) | 此命令设置指定键的值。 |
| 2 | [GET key](http://www.yiibai.com/redis/strings_get.html) | 获取指定键的值。 |
| 3 | [GETRANGE key start end](http://www.yiibai.com/redis/strings_getrange.html) | 获取存储在键上的字符串的子字符串。 |
| 4 | [GETSET key value](http://www.yiibai.com/redis/strings_getset.html) | 设置键的字符串值并返回其旧值。 |
| 5 | [GETBIT key offset](http://www.yiibai.com/redis/strings_getbit.html) | 返回在键处存储的字符串值中偏移处的位值。 |
| 6 | [MGET key1 [key2..]](http://www.yiibai.com/redis/strings_mget.html) | 获取所有给定键的值 |
| 7 | [SETBIT key offset value](http://www.yiibai.com/redis/strings_setbit.html) | 存储在键上的字符串值中设置或清除偏移处的位 |
| 8 | [SETEX key seconds value](http://www.yiibai.com/redis/strings_setex.html) | 使用键和到期时间来设置值 |
| 9 | [SETNX key value](http://www.yiibai.com/redis/strings_setnx.html) | 设置键的值，仅当键不存在时 |
| 10 | [SETRANGE key offset value](http://www.yiibai.com/redis/strings_setrange.html) | 在指定偏移处开始的键处覆盖字符串的一部分 |
| 11 | [STRLEN key](http://www.yiibai.com/redis/strings_strlen.html) | 获取存储在键中的值的长度 |
| 12 | [MSET key value [key value …]](http://www.yiibai.com/redis/strings_mset.html) | 为多个键分别设置它们的值 |
| 13 | [MSETNX key value [key value …]](http://www.yiibai.com/redis/strings_msetnx.html) | 为多个键分别设置它们的值，仅当键不存在时 |
| 14 | [PSETEX key milliseconds value](http://www.yiibai.com/redis/strings_psetex.html) | 设置键的值和到期时间(以毫秒为单位) |
| 15 | [INCR key](http://www.yiibai.com/redis/strings_incr.html) | 将键的整数值增加1 |
| 16 | [INCRBY key increment](http://www.yiibai.com/redis/strings_incrby.html) | 将键的整数值按给定的数值增加 |
| 17 | [INCRBYFLOAT key increment](http://www.yiibai.com/redis/strings_incrbyfloat.html) | 将键的浮点值按给定的数值增加 |
| 18 | [DECR key](http://www.yiibai.com/redis/strings_decr.html) | 将键的整数值减1 |
| 19 | [DECRBY key decrement](http://www.yiibai.com/redis/strings_decrby.html) | 按给定数值减少键的整数值 |
| 20 | [APPEND key value](http://www.yiibai.com/redis/strings_append.html) | 将指定值附加到键 |

.setBit(key, offset, value):key键对应的值value对应的ascii码,在offset的位置(从左向右数)变为value.(此处感谢[@参考文章](http://blog.csdn.net/hgd613/article/details/54095729" \t "_blank)对我理解的帮助),由于二进制只有0和1,此处value只能取0和1,如图,其他值是超出范围的  
.getBit(key, offset):获取键对应值的ascii码的在offset处位值.  
[@ascii码对照表](http://baike.baidu.com/link?url=hFE0BW6q93N2OGtcRV_ynPJ2LejPUatY1tZvKH_oULO04YeQwwJatKduQe6dyeDBJV1s2CFcYoeFPMHGoUnYO_)  
  


Hash:

1.redisTemplate.opsForHash().delete(H key, Object... hashKeys);//...表示可以传入多个map的key，用，隔开。或用数组传值  
2.redisTemplate.opsForHash().hasKey(key, hashKey)；  
3.redisTemplate.opsForHash().get(key, hashKey)；  
4.redisTemplate.opsForHash().entries(key);//返回map集合  
5、6.redisTemplate.opsForHash().increment(H key, HK hashKey, long delta);//或increment(H key, HK hashKey, double delta);；  
7.redisTemplate.opsForHash().keys(key)；//返回map的key集合Set  
8.redisTemplate.opsForHash().size(key)；  
9.redisTemplate.opsForHash().multiGet(H key, Collection<HK> hashKeys);  
10.redisTemplate.opsForHash().putAll(H key, Map<? extends HK, ? extends HV> m)；  
11.redisTemplate.opsForHash().put(key, hashKey, value);  
12.redisTemplate.opsForHash().putIfAbsent(key, hashKey, value)；  
13.redisTemplate.opsForHash().values(key);//返回map中的value集合List；

| **序号** | **命令** | **说明** |
| --- | --- | --- |
| 1 | [HDEL key field2 [field2]](http://www.yiibai.com/redis/hashes_hdel.html) | 删除一个或多个哈希字段。 |
| 2 | [HEXISTS key field](http://www.yiibai.com/redis/hashes_hexists.html) | 判断是否存在散列字段。 |
| 3 | [HGET key field](http://www.yiibai.com/redis/hashes_hget.html) | 获取存储在指定键的哈希字段的值。 |
| 4 | [HGETALL key](http://www.yiibai.com/redis/hashes_hgetall.html) | 获取存储在指定键的哈希中的所有字段和值 |
| 5 | [HINCRBY key field increment](http://www.yiibai.com/redis/hashes_hincrby.html) | 将哈希字段的整数值按给定数字增加 |
| 6 | [HINCRBYFLOAT key field increment](http://www.yiibai.com/redis/hashes_hincrbyfloat.html) | 将哈希字段的浮点值按给定数值增加 |
| 7 | [HKEYS key](http://www.yiibai.com/redis/hashes_hkeys.html) | 获取哈希中的所有字段 |
| 8 | [HLEN key](http://www.yiibai.com/redis/hashes_hlen.html) | 获取散列中的字段数量 |
| 9 | [HMGET key field1 [field2]](http://www.yiibai.com/redis/hashes_hmget.html) | 获取所有给定哈希字段的值 |
| 10 | [HMSET key field1 value1 [field2 value2 ]](http://www.yiibai.com/redis/hashes_hmset.html) | 为多个哈希字段分别设置它们的值 |
| 11 | [HSET key field value](http://www.yiibai.com/redis/hashes_hset.html) | 设置散列字段的字符串值 |
| 12 | [HSETNX key field value](http://www.yiibai.com/redis/hashes_hsetnx.html) | 仅当字段不存在时，才设置散列字段的值 |
| 13 | [HVALS key](http://www.yiibai.com/redis/hashes_hvals.html) | 获取哈希中的所有值 |

List：  
  
redisTemplate.opsForList().leftPush(key, value);//从左向右存压栈  
redisTemplate.opsForList().leftPop(key);//从左出栈  
redisTemplate.opsForList().size(key);//队/栈长  
redisTemplate.opsForList().range(key, start, end);//范围检索,返回List  
redisTemplate.opsForList().remove(key, i, value);//移除key中值为value的i个,返回删除的个数；如果没有这个元素则返回0   
redisTemplate.opsForList().index(key, index);//检索  
redisTemplate.opsForList().set(key, index, value);//赋值  
redisTemplate.opsForList().trim(key, start, end);//裁剪,void,删除除了[start,end]以外的所有元素    
redisTemplate.opsForList().rightPopAndLeftPush(String sourceKey, String destinationKey);//将源key的队列的右边的一个值删除，然后塞入目标key的队列的左边，返回这个值  
注意:要缓存的对象必须实现Serializable接口,因为 Spring 会将对象先序列化再存入 Redis,否则报异常nested exception is java.lang.IllegalArgumentException: DefaultSerializer requires a Serializable……//；；/

| **序号** | **命令** | **说明** |
| --- | --- | --- |
| 1 | [BLPOP key1 [key2 ] timeout](http://www.yiibai.com/redis/lists_blpop.html) | 删除并获取列表中的第一个元素，或阻塞，直到有一个元素可用 |
| 2 | [BRPOP key1 [key2 ] timeout](http://www.yiibai.com/redis/lists_brpop.html) | 删除并获取列表中的最后一个元素，或阻塞，直到有一个元素可用 |
| 3 | [BRPOPLPUSH source destination timeout](http://www.yiibai.com/redis/lists_brpoplpush.html) | 从列表中弹出值，将其推送到另一个列表并返回它; 或阻塞，直到一个可用 |
| 4 | [LINDEX key index](http://www.yiibai.com/redis/lists_lindex.html) | 通过其索引从列表获取元素 |
| 5 | [LINSERT key BEFORE/AFTER pivot value](http://www.yiibai.com/redis/lists_linsert.html) | 在列表中的另一个元素之前或之后插入元素 |
| 6 | [LLEN key](http://www.yiibai.com/redis/lists_llen.html) | 获取列表的长度 |
| 7 | [LPOP key](http://www.yiibai.com/redis/lists_lpop.html) | 删除并获取列表中的第一个元素 |
| 8 | [LPUSH key value1 [value2]](http://www.yiibai.com/redis/lists_lpush.html) | 将一个或多个值添加到列表 |
| 9 | [LPUSHX key value](http://www.yiibai.com/redis/lists_lpushx.html) | 仅当列表存在时，才向列表添加值 |
| 10 | [LRANGE key start stop](http://www.yiibai.com/redis/lists_lrange.html) | 从列表中获取一系列元素 |
| 11 | [LREM key count value](http://www.yiibai.com/redis/lists_lrem.html) | 从列表中删除元素 |
| 12 | [LSET key index value](http://www.yiibai.com/redis/lists_lset.html) | 通过索引在列表中设置元素的值 |
| 13 | [LTRIM key start stop](http://www.yiibai.com/redis/lists_ltrim.html) | 修剪列表的指定范围 |
| 14 | [RPOP key](http://www.yiibai.com/redis/lists_rpop.html) | 删除并获取列表中的最后一个元素 |
| 15 | [RPOPLPUSH source destination](http://www.yiibai.com/redis/lists_rpoplpush.html) | 删除列表中的最后一个元素，将其附加到另一个列表并返回 |
| 16 | [RPUSH key value1 [value2]](http://www.yiibai.com/redis/lists_rpush.html) | 将一个或多个值附加到列表 |
| 17 | [RPUSHX key value](http://www.yiibai.com/redis/lists_rpushx.html) | 仅当列表存在时才将值附加到列表 |

Set：

redisTemplate.opsForValue().getAndSet(key, value)

| **序号** | **命令** | **说明** |
| --- | --- | --- |
| 1 | [SADD key member1 [member2]](http://www.yiibai.com/redis/sets_sadd.html) | 将一个或多个成员添加到集合 |
| 2 | [SCARD key](http://www.yiibai.com/redis/sets_scard.html) | 获取集合中的成员数 |
| 3 | [SDIFF key1 [key2]](http://www.yiibai.com/redis/sets_sdiff.html) | 减去多个集合 |
| 4 | [SDIFFSTORE destination key1 [key2]](http://www.yiibai.com/redis/sets_sdiffstore.html) | 减去多个集并将结果集存储在键中 |
| 5 | [SINTER key1 [key2]](http://www.yiibai.com/redis/sets_sinter.html) | 相交多个集合 |
| 6 | [SINTERSTORE destination key1 [key2]](http://www.yiibai.com/redis/sets_sinterstore.html) | 交叉多个集合并将结果集存储在键中 |
| 7 | [SISMEMBER key member](http://www.yiibai.com/redis/sets_sismember.html) | 判断确定给定值是否是集合的成员 |
| 8 | [SMOVE source destination member](http://www.yiibai.com/redis/sets_smove.html) | 将成员从一个集合移动到另一个集合 |
| 9 | [SPOP key](http://www.yiibai.com/redis/sets_spop.html) | 从集合中删除并返回随机成员 |
| 10 | [SRANDMEMBER key [count]](http://www.yiibai.com/redis/sets_srandmember.html) | 从集合中获取一个或多个随机成员 |
| 11 | [SREM key member1 [member2]](http://www.yiibai.com/redis/sets_srem.html) | 从集合中删除一个或多个成员 |
| 12 | [SUNION key1 [key2]](http://www.yiibai.com/redis/sets_sunion.html) | 添加多个集合 |
| 13 | [SUNIONSTORE destination key1 [key2]](http://www.yiibai.com/redis/sets_sunionstore.html) | 添加多个集并将结果集存储在键中 |
| 14 | [SSCAN key cursor [MATCH pattern] [COUNT count]](http://www.yiibai.com/redis/sets_sscan.html) | 递增地迭代集合中的元素 |

 艾玛，太累了，写着写着发现不用这么麻烦，看到文章中的表了吗？若在java中找不到相应方法就对照表中的一些”字段“在java中找，肯定能找到，而且基本不会浪费多少时间。关于List与Set万能的后人补充吧！

# mongoDB

## 安装教程

<https://www.runoob.com/mongodb/mongodb-window-install.html>

.\mongod --dbpath "D:\var\mongoDB\db" --logpath "D:\var\mongoDB\log\mongodb.log" --install --serviceName "mongodb"

net start mongodb

\mongod --dbpath "D:\var\mongoDB\db" --logpath "D:\var\mongoDB\log\mongodb.log" --auth

## springboot mongoDB

pegdown是把markdown文件转成html文件

**Spring HATEOAS** 是一个用于**支持实现超文本驱动**的**REST Web 服务的开发库**。是 [HATEOAS](http://en.wikipedia.org/wiki/HATEOAS) 的实现。

（HATEOAS背后的思想就是**响应中包含指向其它资源的链接**。**客户端可以利用这些链接和服务器交互**）

参考资料: <https://www.cnblogs.com/kaixinyufeng/p/8283289.html>

restful web

jackson-annotations json解析