Redis

1 Redis 入门

1.1 简介

Redis 是一个开放源码（BSD 许可）、内存中的数据结构存储，用作数据库、缓存和 message broker。它支持数据结构，如字符串、散列、列表、集合、带有范围查询的排序集、位图、超 loglogs 和带有半径查询的地理空间索引。Redis 有内置的复制、Lua 脚本、LRU 驱逐、交易和不同级别的磁盘持久性，并通过 Redis 哨兵和 Redis 集群自动分区提供高可用性。

Redis 是一个高性能的 KEY-VALUE（内存）数据库。

1.2 优势

与其他 key - value 缓存产品有以下三个特点：

- redis 支持数据的持久化，可以将内存中的数据保存在磁盘中，重启的时候再次加载进行使用。
- redis 不仅仅支持 key-value 类型的数据，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。

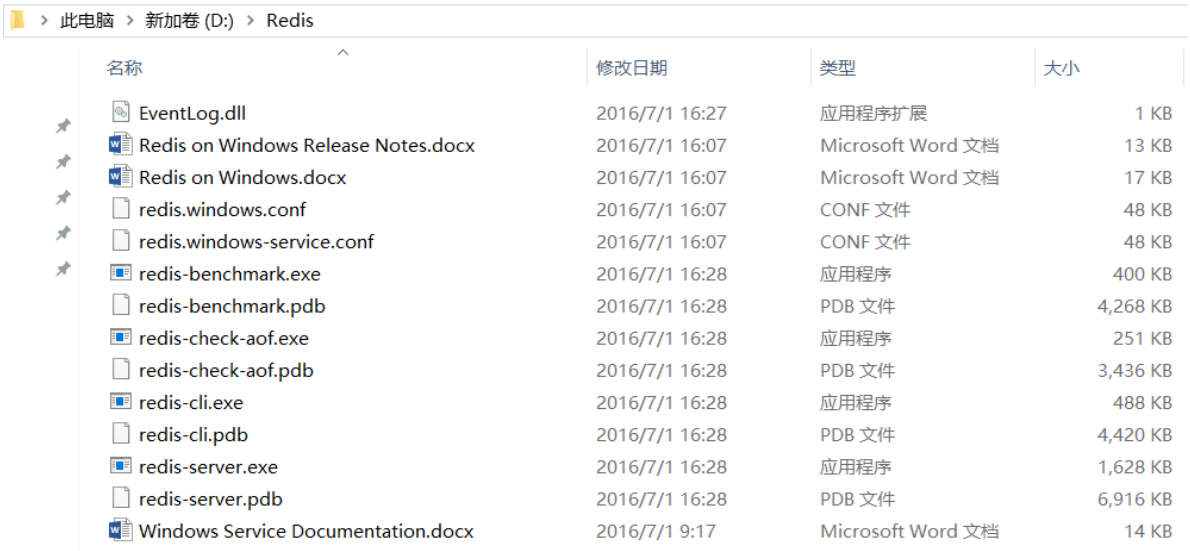
redis 自身优势

- **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。
- **原子性** – Redis 的所有操作都是原子性的，通过 MULTI 和 EXEC 指令包起来。
- **丰富的特性** – Redis 还支持 publish/subscribe, advice, key 过期等等特性。

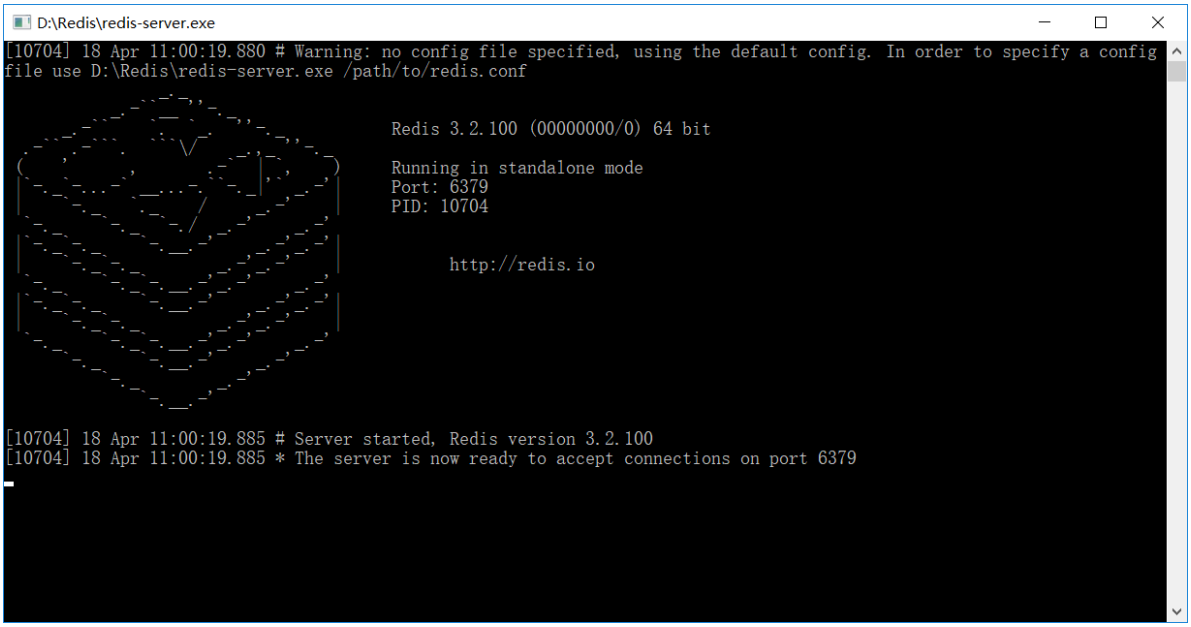
2 Redis 安装

2.1 Windows

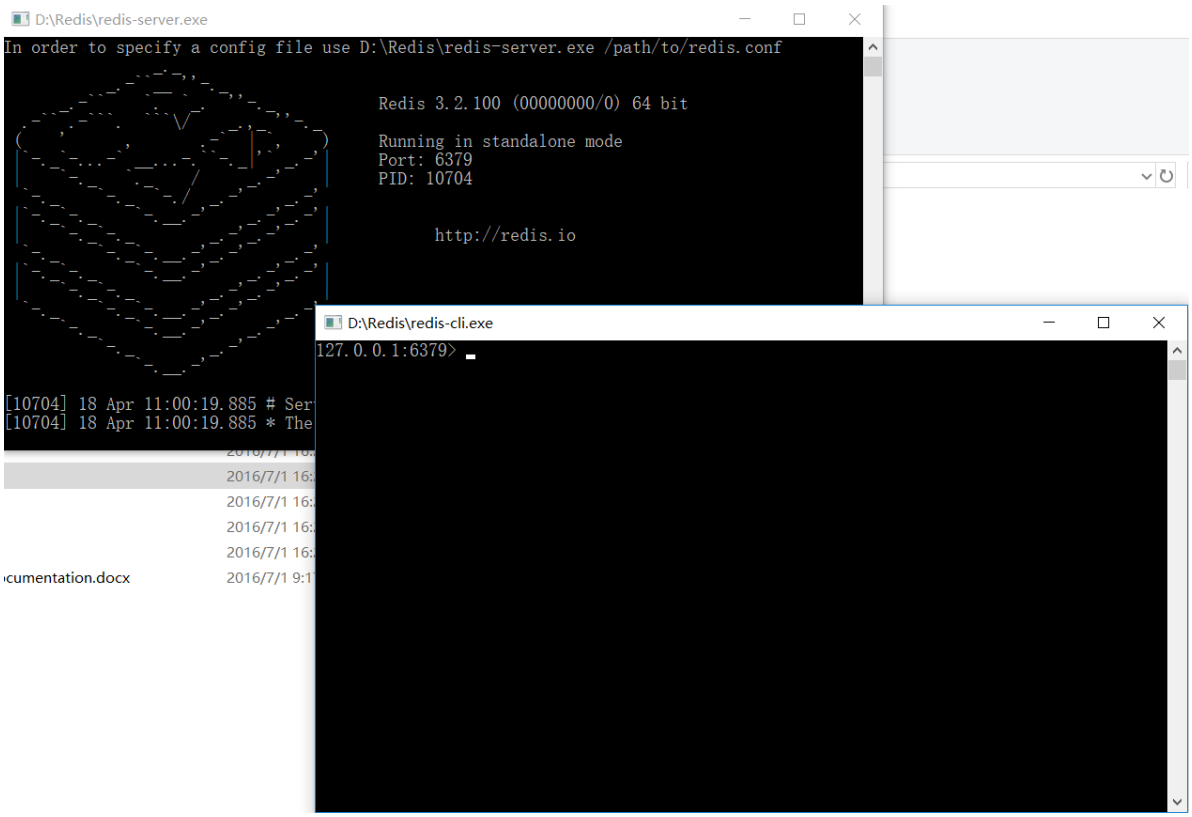
Redis 的 Windows 版本区分 32、64bit 注意操作系统的版本号。



例如解压到 D:\Redis 目录下（如上图）



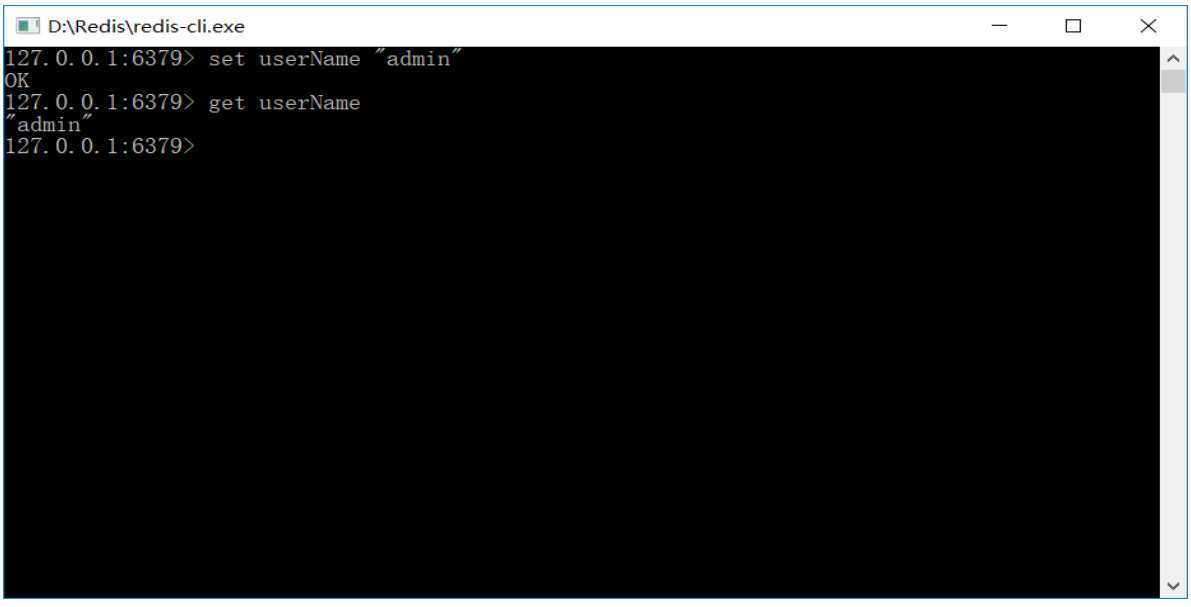
双击打开 redis-server.exe(如上图)



双击打开 redis-cli.exe **不要关闭** **不要关闭** **不要关闭** redis-server.exe(如上图)

redis-server.exe 属于 redis 服务器，首先要保证的是服务器端开启状态， 然后再次打开 redis-cli.exe 属于 redis 客户端。

127.0.0.1:6379> 127.0.0.1 属于 IP 地址， 6379 属于端口号。



2.2 Linux

```
# 1.下载 linux 版本 redis
$ wget http://download.redis.io/releases/redis-4.0.7.tar.gz

# 2.解压 redis 压缩文件
$ tar xzf redis-2.8.17.tar.gz

# 3.进入到 redis 解压目录
$ cd redis-4.0.7

# 4.编译安装
$ make

# 5.启动 redis 服务端
$ ./redis-server

# 6.启动 redis 客户端
$ ./redis-cli

      redis 127.0.0.1:6379 > set username “admin”

      OK

      redis 127.0.0.1:6379 > get username

      “admin”
```

3 Redis 数据类型

redis 支持五种数据类型：

1. string（字符串）
2. hash（哈希）
3. list（列表）
4. set（集合）
5. zset(sorted set：有序集合)

3.1 String · 字符串

redis 中的 String 类型属于最基本的数据类型， 它是由 key-value 键值对构成， String 的类型是二进制安全的， redis 的 String 可以包含任意的数据字符串，数字， 布尔或文件、序列化后的对象。redis 中的 String 每个 key 最大存储量为 512M。

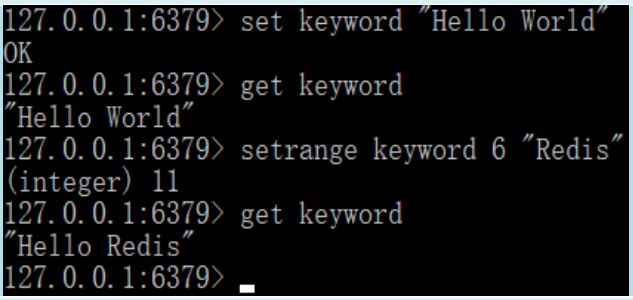
3.1.1 示例

```
#String 类型存放示例

127.0.0.1:6379> set name admin
OK
127.0.0.1:6379> get name
"admin"
127.0.0.1:6379> set age 25
OK
127.0.0.1:6379> get age
"25"
127.0.0.1:6379> set flag true
OK
127.0.0.1:6379> get flag
"true"
```

3.1.2 命令

序号	命令	描述
1	**set key value**	为指定的 key 设置 value 值
2	**get key**	根据指定的 key 获取 value 值

3	**getset key value**	替换指定 key 的 value 值并返回旧值
4	**getrange key start end**	根据指定的 key 获取截取区间值
5	**getbit key offset**	获取指定 key 的二级制位置数字 例如 key=a 在redis中会以 a = 97 = 01100001 这样的二进制存储 offset 1 = 0 offset 2 = 1 offset 3 = 1
6	**setbit key offset value**	修改指定 key 的二进制位置数字
7	**mget key key1 key2**	获取一个或多个指定 key 的值
8	**mset key value key1 values...**	设置一个或多个 key-value 值
9	**setex key seconds value**	将 value 值关联到 key 并设置过期日期(秒)
10	**psetex key milliseconds value**	将 value 值关联到 key 并设置过期日期(毫秒)
11	**setnx key value**	只有不存在 key 可以调用并设置值
12	**msetnx key value key1 value1**	只有不存在 key 可以调用并设置值（多个）
13	**setrange key offset value**	
14	**strlen key**	获取指定 key 存储 value 的长度
15	**incr key**	指定 key 的值+1
16	**incrby key increment**	指定 key 的值 +? （?为指定量）
17	**incrbyfloat key increment**	指定 key 的值 +? （?为浮点型）
18	**decr key**	指定 key 的值-1

19	decrby key decrement	指定 key 的值 -? （?为指定量）
20	append key value	向指定 key 的 value 后追加值（拼接）

3.2 Hash·哈希

redis 中的 Hash 类型属于 key-value 键值对构成的集合（与 String 区分）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。

每个 hash 可以存储 $2^{32}-1$ 键值对（4294967295）

3.2.1 示例

```
127.0.0.1:6379> hmset user name "admin" age 25 address "beijing"
OK
127.0.0.1:6379> hgetall user
1) "name"
2) "admin"
3) "age"
4) "25"
5) "address"
6) "beijing"
127.0.0.1:6379> hget user name
"admin"
127.0.0.1:6379>
```

3.2.2 命令

序号	命令	描述
1	hmset key field value field1 value1	向哈希集合中添加 key 并设置指定属性与值
2	hgetall key	输出指定 key 中的所有字段及字段值
3	hset key field value	替换指定 key 的 field 对应 value 值，不存在则创建
4	hmget key field field1...	获取指定 key 的 filed 字段值

5	hget key field	获取指定 key 中的指定 field 属性值
6	hdel key field field1	删除指定 key 中的字段
7	hexists key field	判断指定 key 中 field 是否存在 (0:不存在 1:存在)
8	hlen key	返回指定 key 中 field 数量
9	hkeys key	返回指定 key 中所有 field 字段名称
10	hsexnx key field value	向指定 key 中添加不存在的属性值（不能存在）
11	hvals key	获取指定 key 中所有的 value 值
12	hincrby key field increment	指定 key 中 field 字段值+? (?指定的数字值)
13	hincrbyfloat key field increment	指定 key 中 field 字段值+? (?指定的浮点型)

4 Spring-Redis 整合

4.1 pom.xml

由 Maven 管理的项目中 pom.xml 负责管理 jar， 当我们需要使用 Spring 框架与 Redis 内存数据库整合的时候， 需要有 pom.xml 管理 jar 包。

```
#配置版本号#
<redis.version>2.9.0</redis.version>
<spring.data.version>2.0.6.RELEASE</spring.data.version>


#引入相关文件#
<dependency>
      <groupId>org.springframework.data</groupId>
      <artifactId>spring-data-redis</artifactId>
      <version>${spring.data.version}</version>
</dependency>
<dependency>
      <groupId>redis.clients</groupId>
      <artifactId>jedis</artifactId>
      <version>${redis.version}</version>
</dependency>
```

4.2 redis.properties

```
redis.host=127.0.0.1
redis.port=6379
#redis.pass=password
redis.pooled=true
redis.dbIndex=0
redis.expiration=3000
redis.maxIdle=300
redis.maxActive=600
redis.maxWait=1000
redis.testOnBorrow=true
```

4.3 applicationContext.xml

```
    <!—redis 连接池-->
    <bean id="jedisPoolConfig" class="redis.clients.jedis.JedisPoolConfig">
            <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="jedisConnectionFactory" class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory">
            <property name="hostName" value="${redis.host}"></property>
            <property name="port" value="${redis.port}"></property>
            <property name="usePool" value="${redis.pooled}"></property>
            <property name="poolConfig" ref="jedisPoolConfig"></property>
            <property name="database" value="${redis.dbIndex}" />
    </bean>

    <!—redis 操作模板-->
    <bean id="redisTemplate" class="org.springframework.data.redis.core.RedisTemplate">
          <property name="connectionFactory" ref="jedisConnectionFactory"></property>
          <property name="keySerializer">
                  <bean class="org.springframework.data.redis.serializer.StringRedisSerializer"></bean>
           </property>
           <property name="valueSerializer">
                  <bean class="org.springframework.data.redis.serializer.StringRedisSerializer"></bean>
            </property>
            <property name="hashKeySerializer">
                  <bean class="org.springframework.data.redis.serializer.StringRedisSerializer"></bean>
            </property>
    </bean>
```

4.4 RedisTemplate

```
    @Autowired
    private RedisTemplate<Serializable, Serializable> redisTemplate;

    @Test
    public void test() {
```

```
        // TODO Auto-generated method stub
        //通过模板获取操作 String 对象
        ValueOperations<Serializable, Serializable> value = redisTemplate.opsForValue();

        //通过模板获取操作 Hash 对象
        HashOperations<Serializable, Object, Object> hash = redisTemplate.opsForHash();

        //通过模板获取操作 List 对象
        ListOperations<Serializable, Serializable> list = redisTemplate.opsForList();

        //通过模板获取操作 Set 对象
        SetOperations<Serializable, Serializable> set = redisTemplate.opsForSet();

        //通过模板获取操作 ZSet 对象
        ZSetOperations<Serializable, Serializable>  zset = redisTemplate.opsForZSet();
    }
```

5 BSD 许可

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。