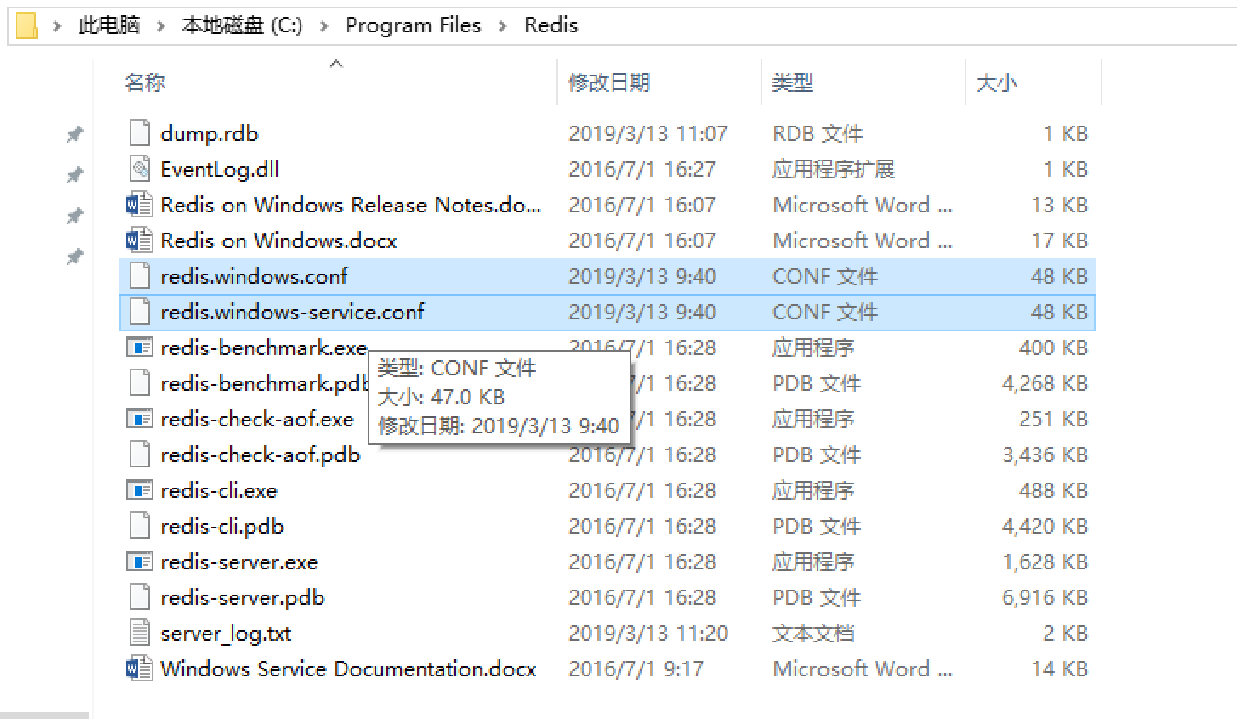
Redis数据库:

学习网站:http://www.runoob.com/redis/redis-sorted-sets.html

一.安装Redis数据库:

需要改动以下文件,修改密码(大约在440多行)



› 此电脑 › 本地磁盘 (C:) › Program Files › Redis

名称	修改日期	类型	大小
dump.rdb	2019/3/13 11:07	RDB 文件	1 KB
EventLog.dll	2016/7/1 16:27	应用程序扩展	1 KB
Redis on Windows Release Notes.do...	2016/7/1 16:07	Microsoft Word ...	13 KB
Redis on Windows.docx	2016/7/1 16:07	Microsoft Word ...	17 KB
redis.windows.conf	2019/3/13 9:40	CONF 文件	48 KB
redis.windows-service.conf	2019/3/13 9:40	CONF 文件	48 KB
redis-benchmark.exe	2016/7/1 16:28	应用程序	400 KB
redis-benchmark.pdb	/1 16:28	PDB 文件	4,268 KB
redis-check-aof.exe	/1 16:28	应用程序	251 KB
redis-check-aof.pdb	2016/7/1 16:28	PDB 文件	3,436 KB
redis-cli.exe	2016/7/1 16:28	应用程序	488 KB
redis-cli.pdb	2016/7/1 16:28	PDB 文件	4,420 KB
redis-server.exe	2016/7/1 16:28	应用程序	1,628 KB
redis-server.pdb	2016/7/1 16:28	PDB 文件	6,916 KB
server_log.txt	2019/3/13 11:20	文本文档	2 KB
Windows Service Documentation.docx	2016/7/1 9:17	Microsoft Word ...	14 KB

类型: CONF 文件
大小: 47.0 KB
修改日期: 2019/3/13 9:40

二.启动或重启Redis:services.msc(修改密码后重启)

三.在cmd中操作Redis数据库

Redis数据库数据库分为以下几种类型:是一个由Salvatore Sanfilippo写的key-value存储系统,它通常被称为数据结构服务器，因为值（value）可以是 字符串(String), 哈希(Hash), 列表(list), 集合(sets) 和 有序集合(sorted sets)等类型。

```
C:\Users\lanou>redis-cli
127.0.0.1:6379> auth 123456
OK
##字符串
127.0.0.1:6379> set lanou zzj181105;
```

```
OK
127.0.0.1:6379> get lanou
"zzj181105;"
127.0.0.1:6379> set number 10000
OK
##自增1
127.0.0.1:6379> incr number
(integer) 10001
127.0.0.1:6379> del number
(integer) 1
127.0.0.1:6379> get number
(nil)
```

##哈希值

```
127.0.0.1:6379> hmset student name xiaohong age 18 sex man
OK
##查看全部
127.0.0.1:6379> hgetall student
1) "name"
2) "xiaohong"
3) "age"
4) "18"
5) "sex"
6) "man"
##查看单个
127.0.0.1:6379> hget student name
"xiaohong"
```

##redis(list列表),有序可重复

```
##将一个值插入到已存在的列表头部,最后插入的在最前面
127.0.0.1:6379> lpush cuntry USA CHINA UK
(integer) 3
127.0.0.1:6379> LRANGE cuntry 0 10
```

```
1) "UK"
2) "CHINA"
3) "USA"
##之前插入
127.0.0.1:6379> linsert coutry before USA HK
(integer) 1
127.0.0.1:6379> lrange cuntry 0 10
1) "UK"
2) "CHINA"
3) "HK"
4) "USA"
##该list的长度,获取长度用size,长度减一,最大下标
127.0.0.1:6379> llen cuntry
(integer) 4
127.0.0.1:6379> lindex cuntry 2
"UK"
```

##redis(set集合):不可重复,String的无序集合

```
127.0.0.1:6379> sadd city bejing zhengzhou shanghai newyerk beijing
(integer) 5
#输出无序
127.0.0.1:6379> smembers city
1) "shanghai"
2) "newyerk"
3) "zhengzhou"
4) "bejing"
5) "beijing"
127.0.0.1:6379> smembers city
1) "shanghai"
2) "newyerk"
3) "zhengzhou"
4) "bejing"
5) "beijing"
```

#Redis 有序集合(sorted set):1.string类型元素的集合,且不允许重复的成员。2.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(升序),值相同覆盖.

```
127.0.0.1:6379> zadd class 1 string
(integer) 1
127.0.0.1:6379> zadd class 1.2 Date
(integer) 1
127.0.0.1:6379> zadd class 1.1 Date
(integer) 1
127.0.0.1:6379> zadd class 1.2 SimpleTimeFormat
(integer) 1
127.0.0.1:6379> zadd class 1.2 Calender
(integer) 1
##范围
127.0.0.1:6379> zrange class 0 10
1) "string"
2) "Date"
3) "Calender"
4) "SimpleTimeFormat"
127.0.0.1:6379> zrange class 0 10 withscores
1) "string"
2) "1"
3) "Date"
4) "1.1000000000000001"
5) "Calender"
6) "1.2"
7) "SimpleTimeFormat"
8) "1.2"
```

四.Redis数据库与spring-boot的集成:

```
# 端口号
server:
 port: 9999
```

```

#datasouce数据源
spring:
 datasource:
 type: com.alibaba.druid.pool.DruidDataSource
 username: root
  password: root
  driver-class-name: com.mysql.jdbc.Driver
  url: jdbc:mysql://localhost:3306/shop
  dbcp2: # 进行数据库连接池的配置
  min-idle: 5 # 数据库连接池的最小维持连接数
  initial-size: 5 # 初始化提供的连接数
  max-total: 5 # 最大的连接数
  max-wait-millis: 200 # 等待连接获取的最大超时时间
  redis:
  database: 0
  host: 127.0.0.1
  password: 123456
  port: 6379
  timeout: 10000
  jedis:
  pool:
  max-active: 8 # 连接池最大连接数（使用负值表示没有限制）
  max-wait: -1 # 连接池最大阻塞等待时间（使用负值表示没有限制）
  max-idle: 8 # 连接池中的最大空闲连接
  min-idle: 0 # 连接池中的最小空闲连接
 #mybatis
 mybatis:
  mapper-locations: classpath:mapping/*.xml
  type-aliases-package: com.example.demo.domain
 #log4j
 logging:
  level:
  com.example.demo.domain: debug
```

generatorConfiguration:(自动生成.注意包名,工程名)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN" "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">
<generatorConfiguration>
 <!-- 驱动的绝对位置 -->
 <classPathEntry
 location="E:\Maven\mysql\mysql-connector-java\5.1.39\mysql-connector-java-5.1.39.jar" />
 <context id="context1">
 <!-- 去掉注释 -->
 <commentGenerator>
  <property name="suppressDate" value="true"/>
  <property name="suppressAllComments" value="true" />
  </commentGenerator>
  <jdbcConnection
  connectionURL="jdbc:mysql://localhost:3306/shop"
  driverClass="com.mysql.jdbc.Driver" password="root" userId="root" />
  //包名:domain下,文件下:src/main/java
  <javaModelGenerator
  targetPackage="com.lanou.springboot_demo02.domain"
  targetProject="springboot_demo02/src/main/java" />
  //包名:mapping,文件下:src/main/resources
  <sqlMapGenerator targetPackage="mapping"
  targetProject="springboot_demo02/src/main/resources" />
  <javaClientGenerator
  targetPackage="com.lanou.springboot_demo02.domain"
  targetProject="springboot_demo02/src/main/java"
  type="XMLMAPPER" />
  <table tableName="smbms_user" domainObjectName="User"
  enableCountByExample="false" enableUpdateByExample="false"
  enableDeleteByExample="false" enableSelectByExample="false"
  selectByExampleQueryId="false">
  <!-- 属性的驼峰的设置 -->
  <property name="useActualColumnNames" value="true" />
  </table>
  </context>
</generatorConfiguration>
```

controller:

spring-boot-resuful

```
@RestController
@RequestMapping("user")
@Api(value="用户请求")
public class UserController {

 @Resource
 private UserService userService;
 //说明
 @ApiOperation(value="获取所有用户信息",notes="获取所有用户信息")
  @RequestMapping(value="/",method=RequestMethod.GET)
  public Object getUser(){
  //当前页内容
  //get请求在地址栏可以直接测出来,put,delete,post需要借助swagger
  List<User> users=userService.page(1,10).getList();
  return new Result(200,null,users);

  }
  @ApiOperation(value="通过id获取所有用户信息",notes="通过id获取所有用户信息")
  @RequestMapping(value="{id}",method=RequestMethod.GET)
  public Object getUserById(
  @ApiParam(name="id",value="用户id",required=true)
  @RequestParam Long id){
  User u=userService.findUserById(id);
  return new Result(200,null,u);
  }

  //传参是json
  //@RequestBody中添加@ApiParam自动将Json转为对象
  //简写
```

```java
@ApiOperation("新增用户信息")
@PostMapping(value="/")
public Object insertUser(
@RequestBody @ApiParam(value="用户信息") User u) {

int row=userService.insert(u);
if (row>0) {
return new Result(200,null,null);
}else {
return new Result(500,"添加失败",null);

}

}
@PutMapping(value="{id}")
@ApiOperation(value="根据id修改用户信息")
public Object update(
@ApiParam(name="id",value="用户Id",required=true)
@RequestParam Long id,
@RequestBody @ApiParam(value="用户信息") User u) {
//获取模板参数
u.setId(id);
int row=userService.updata(u);
if (row>0) {
return new Result(200,null,null);
}else {
return new Result(500,"修改失败",null);
}
}
@DeleteMapping("{id}")
@ApiOperation(value="根据id删除用户信息")
public Object deleteUser(
@ApiParam(name="id",value="用户Id",required=true)
@RequestParam Long id) {
int row=userService.deleteById(id);
```

```
 if (row>0) {
 return new Result(200,null,null);
 }else {
 return new Result(500,"删除失败",null);
 }
 }
}
```