**Redis入门文档**

[一、Redis概念 1](#_Toc1369)

[二、 Redis安装使用 1](#_Toc7351)

[2.1 redis安装 1](#_Toc27049)

[2.2 Springboot整合redis缓存 1](#_Toc22434)

[2.2.1操作redisTemplate 2](#_Toc9593)

[2.2.2缓存注解 2](#_Toc29217)

[2.2.2.1@CacheConfig 2](#_Toc12623)

[2.2.2.2@Cacheable 2](#_Toc27972)

[2.2.2.3@CachePut 2](#_Toc21016)

[2.2.2.4@CacheEvict 3](#_Toc2551)

[2.2.2.5@Caching 3](#_Toc24053)

[2.3redis集群 4](#_Toc8982)

**一、Redis概念**

Redis是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库

支持存储的value类型，包括string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和hash（哈希类型）

1. **Redis安装使用**

**2.1 redis安装**

默认端口6379

**2.1.1Linux安装**

Redis因为C编写的需要下载源代码编译

Linux安装 环境cenos7

官网地址

<https://redis.io/download>

选择稳定版下载压缩包

然后通过xftp上传到linux文件下/usr/local/redis

我下载的是4.0.14的

解压文件 tar -zxvf redis-4.0.14.tar.gz

编译前需要安装gcc

yum install -y gcc

到解压后的文件夹 执行make

编译完成后 redis.conf为redis配置

/usr/local/redis/redis-4.0.14/src 下

redis-server和redis-cli分别为服务端和客户端

直接执行redis-server 启动redis 可以另起redis-cli客户端操作redis新增查询修改数据等。

配置文件

Requirepass 设置密码123456

Requirepass 123456

Daemonize 后台运行

Daemonize yes

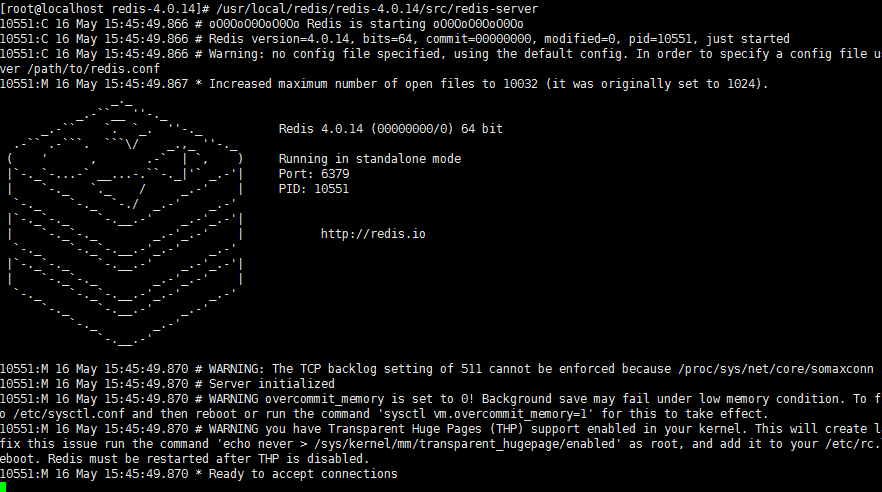
bind protected-mode 允许外部访问

注释 bind 127.0.0.1

protected-mode no

并且关闭该端口的防火墙重启redis即可

/usr/local/redis/redis-4.0.14/src/redis-server /usr/local/redis/redis-4.0.14/redis.conf



**2.1.2Windows安装**

**Redis项目不正式支持Windows。 但是，Microsoft开放技术小组开发和维护这个Windows端口针对Win64。**

下载地址

<https://github.com/microsoftarchive/redis/releases>

解压文件 在当前目录打开运行redis-server redis.windows.conf

配置文件修改与linux相同

**2.2 Springboot整合redis缓存**

**主方法加注释**

**@EnableCaching**

@EnableCaching注解是spring framework中的注解驱动的缓存管理功能。自spring版本3.1起加入了该注解。如果你使用了这个注解，那么你就不需要在XML文件中配置cache manager了。

当你在配置类(@Configuration)上使用@EnableCaching注解时，会触发一个post processor，这会扫描每一个spring bean，查看是否已经存在注解对应的缓存。如果找到了，就会自动创建一个代理拦截方法调用，使用缓存的bean执行处理。

**pom.xml**

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-redis</artifactId>  
</dependency>  
  
<dependency>  
 <groupId>redis.clients</groupId>  
 <artifactId>jedis</artifactId>  
</dependency>

Jedis redis的java客户端 用于封装调用服务端的api springboot默认使用Lettuce作为redis客户端

**配置文件**

# Redis数据库索引（默认为0）

spring.redis.database=0

# Redis服务器地址

spring.redis.host=127.0.0.1

# Redis服务器连接端口

spring.redis.port=6379

# Redis服务器连接密码（默认为空）

spring.redis.password=

# 连接池最大连接数（使用负值表示没有限制）

spring.redis.pool.max-active=200

# 连接池最大阻塞等待时间（使用负值表示没有限制）

spring.redis.pool.max-wait=-1

# 连接池中的最大空闲连接

spring.redis.pool.max-idle=10

# 连接池中的最小空闲连接

spring.redis.pool.min-idle=0

# 连接超时时间（毫秒）

spring.redis.timeout=1000

**2.2.1操作redisTemplate缓存**

需要自己处理业务逻辑

完成上述安装配置后就可以直接通过注入redisTemplate操作redis了

spring-redis对redis的五种数据类型也有支持

HashOperations：对hash类型的数据操作

ValueOperations：对redis字符串类型数据操作

ListOperations：对链表类型的数据操作

SetOperations：对无序集合类型的数据操作

ZSetOperations：对有序集合类型的数据操作

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

示例

@Autowired

RedisTemplate redisTemplate;

redisTemplate.opsForValue().set("test","testValue");//根据key设置缓存

redisTemplate.opsForValue().get("test");//根据key获取缓存

redisTemplate.delete("test");//根据key删除缓存

redisTemplate.getExpire("test")//根据key获取过期时间

更多使用方法请查看官方Api

**2.2.2缓存注解**

**2.2.2.1@CacheConfig**

概述:

基于类级别的缓存注解,允许共享缓存的名称、KeyGenerator、CacheManager 和CacheResolver

**2.2.2.2@Cacheable**

概述:

声明方法是可缓存的,每次执行方法前判断是否已有缓存 如果有就直接返回，不执行从数据库操作。

cacheNames:

存储注解方法调用结果的缓存名称。

key：

缓存的 key，可以为空（建议指定）。如果指定要按照 SpEL 表达式

编写，如果不指定，则默认按照方法的所有参数进行组合。可以通过redisconfig配置生成值

condition：

缓存的条件，可以为空，使用 SpEL 编写，返回boolean，只有为 true 才进行缓存

示例:

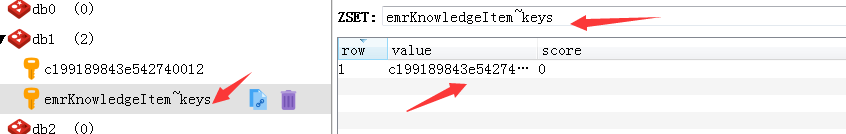
@Cacheable(cacheNames="emrKnowledgeItem" ,key="#p0") key为第一个参数 或者key="#id"根据参数名获取

public EmrKnowledgeItem getByPk(String id)

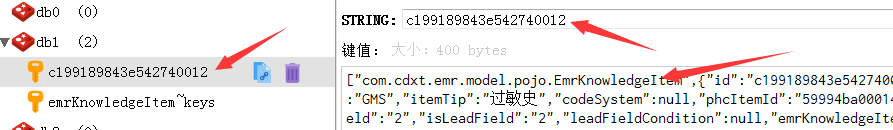
Redis 根据key-value存储 key首先存放cacheNames为emrKnowledgeItem value为方法定义的key这里就是id

再根据映射 key为id value为getByPk返回值也就是EmrKnowledgeIte实体

第一个映射



第二个映射



**2.2.2.3@CachePut**

概述:

根据方法的请求参数对其结果进行缓存,每次都会执行该方法，并将执行结果以键值对的形式存入指定的缓存中.适用于update的时候

cacheNames:

存储注解方法调用结果的缓存名称。

key：

缓存的 key，可以为空（建议指定）。如果指定要按照 SpEL 表达式

编写，如果不指定，则默认按照方法的所有参数进行组合。

cacheManager：

指定使用的cacheManager。

condition：

缓存的条件，可以为空，使用 SpEL 编写，返回 true 或者

false，只有为 true 才进行缓存

示例:

@CachePut(cacheNames="book", key="#isbn")

public Book updateBook(ISBN isbn, BookDescriptor descriptor)

**2.2.2.4@CacheEvict**

概述:

指定一个或多个缓存，使之都受影响。提供了一个额外的参数allEntries 。表示是否需要清除缓存中的所有元素。默认为false，表示不需要。当指定了allEntries为true时，Spring Cache将忽略指定的key,指定cacheNames下的缓存全部失效。适用于delete的时候

该注解除了和其他一样的cacheNames、condition

还多了allEntries、beforeInvocation

beforeInvocation属性 用于在方法执行前就删除缓存

@CacheEvict(value="knowledgeList",condition="#record.isDatasource.equals('2')",allEntries=true)

public int deleteByPk(String id)

执行方法后直接清除满足condition条件的名为knowledgeList的所有缓存

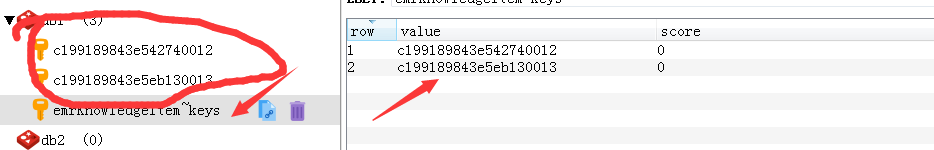
@CacheEvict(value = "emrKnowledgeItem", key = "#p0",,allEntries=false)

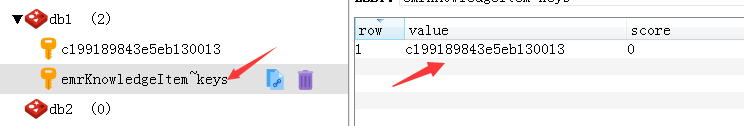
public int deleteByPk(String id)

执行方法后直接清除emrKnowledgeItemkey为id相同的缓存

allEntries为true直接删除满足条件名为emrKnowledgeItem的所有缓存

allEntries为false删除满足条件key为id的缓存





**2.2.2.5@Caching**

概述:

是Cacheable、CachePut、CacheEvict的组合注解可以看成一个容器可以将多个缓存注解放入。

示例

@Caching(

**evict** = {

@CacheEvict(value="knowledgeList",condition="#record.isDatasource.equals('2')",allEntries= true),

@CacheEvict(value="dataSourceList",condition="#record.isDatasource.equals('1')",allEntries=true)

},

cacheable = {

@Cacheable(cacheNames="emrKnowledgeItem" ,key="#p0")

},

put ={

@CachePut(cacheNames="book", key="#id")

})

public int deleteByPk(String id)

**2.3redis集群**

1.主从复制 主负责读写 从负责读 主写完成后会自动同步到从

2.哨兵 一个主多个哨 一个哨只能1个主 主要用于发现主数据库是否宕机 宕机时通过投票将从变为主

3.集群 分布式存储 将所有内存根据一定算法分配给各个redis 至少需要3主3从 根据主分配 主宕机后 从自动提升 主从都挂了 集群不可用

**redis安装**

官网下载redis 下载3以上版本 只有3以上版本支持cluster集群配置

1.新建目录 usr/local/redis 并将压缩文件传到linux

2.解压文件 tar -zxvf redis-4.0.14.tar.gz

tar -zxvf 6385.tar.gz

3.cd 到解压文件 执行 ./configure 默认配置 再执行make sudo make install 完成redis安装

配置文件启动执行

/usr/local/redis/redis-4.0.14/src/redis-server /usr/local/redis/redis-4.0.14/redis.conf

cluster-enabled yes 开启集群

Requirepass  设置密码123456

Requirepass 123456

Daemonize   后台运行

Daemonize yes

bind   protected-mode 允许外部访问

注释 bind 127.0.0.1

protected-mode  no

**启动redis**

/usr/local/redis/redis-4.0.14/src/redis-server /usr/local/redis/redis-4.0.14/redis.conf

开放防火墙端口

并且关闭该端口的防火墙重启redis即可

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

firewall-cmd --permanent --zone=public --add-port=6380/tcp

。。。

**安装基础依赖**

* yum update
* yum -y install make gcc gcc-c++ kernel-devel

**安装ruby**

要2.2.0以上版本

yum -y install wget

wget <https://cache.ruby-lang.org/pub/ruby/2.3/ruby-2.3.7.tar.gz>

tar xvzf ruby-2.3.7.tar.gz -C /usr/local/redis

cd /usr/local/redis/ruby-2.3.7

./configure --with-openssl-dir=/usr/local/ssl

make && make install

ruby -v

**安装 zlib-devel**

* yum install zlib-devel
* cd /usr/local/redis/ruby-2.3.7/ext/zlib
* ruby ./extconf.rb
* make && make install

**安装 openssl-devel**

* cd /usr/local/redis/ruby-2.3.7/ext/openssl
* ruby ./extconf.rb
* ln -s /root/redis/ruby-2.3.7/include /
* 如果提示已经有了则移除再执行 rm /include
* ln -s /root/redis/ruby-2.3.7/include /
* make && make install

**安装 redis接口**

* gem install redis

 错误时

ERROR: While executing gem ... (Gem::Exception) Unable to require openssl, install OpenSSL and rebuild ruby (preferred) or use non-HTTPS sources

出现openssl错误 编辑ruby下的

/usr/local/redis/ruby-2.3.7/ext/openssl Makefile 所有 $(top\_srcdir) 为 ../..

再次执行

**启动redis**

启动所有redis 至少6个redis

到任意一个redis src目录下有redis-trib.rb文件

执行

./redis-trib.rb create --replicas 1 172.16.206.130:6379 172.16.206.130:6380 172.16.206.130:6385 172.16.206.130:6381 172.16.206.130:6382 172.16.206.130:6383

**完成集群配置**

如果启动失败请注意配置redis.conf

cluster-enabled yes 开启集群

Requirepass  设置密码123456

Requirepass 123456

Daemonize   后台运行

Daemonize yes

bind   protected-mode 允许外部访问

注释 bind 127.0.0.1

protected-mode  no

并且关闭该端口的防火墙重启redis即可

cluster-config-file nodes-6380.conf 集群时配置文件

cluster-node-timeout 15000