**# flood-cache**

**## 简介**

基于redis 及本地的Caffeine二级缓存的服务。

<br>

**## 导入**

**\*\*SpringBoot项目导入：\*\***

```xml

<dependency>

<groupId>cn.flood</groupId>

<artifactId>flood-cache</artifactId>

<version>2.0.0</version>

</dependency>

```

**\*\*jar包方式导入：\*\***

<br>

**## 用法**

**### 在SpringBoot中使用**

在`application.properties`有如下配置可选：

**########################################################**

**### redis setting. 2.0以后redis的连接池改为了lettuce，lettuce能够支持redis4，需要java8及以上**

**########################################################**

**#单机配置**

**#---**

**#spring:**

**# redis:**

**# password: winsunion123**

**# database: 0 #Redis默认情况下有16个分片，这里配置具体使用的分片，默认是0**

**# timeout: 10000 # 数据库连接超时时间，2.0 中该参数的类型为Duration，这里在配置的时候需要指明单位**

**# #单机配置**

**# host: 10.10.10.51**

**# port: 6379**

**# # 连接池配置，2.0中直接使用jedis或者lettuce配置连接池**

**# lettuce:**

**# pool:**

**# # 连接池最大活跃连接数（使用负值表示没有限制） 默认 8,负数为不限制**

**# max-active: 20**

**# # 连接池中的最大空闲连接 默认 8**

**# max-idle: 20**

**# # 最小空闲连接数**

**# min-idle: 15**

**# # 连接池最大阻塞等待时间（使用负值表示没有限制） 默认 -1**

**# max-wait: 30000**

**#集群配置**

**---**

spring:

redis:

password:

encode: utf-8

database: 0 #Redis默认情况下有16个分片，这里配置具体使用的分片，默认是0

timeout: 10s # 数据库连接超时时间，2.0 中该参数的类型为Duration，这里在配置的时候需要指明单位

cluster:

max-redirects: 3

nodes:

- 47.94.7.243:6001

- 47.94.7.243:6002

- 47.94.7.243:6003

- 47.94.7.243:6004

- 47.94.7.243:6005

- 47.94.7.243:6006

# 连接池配置，2.0中直接使用jedis或者lettuce配置连接池

lettuce:

pool:

# 连接池最大活跃连接数（使用负值表示没有限制） 默认 8,负数为不限制

max-active: 20

# 连接池中的最大空闲连接 默认 8

max-idle: 20

# 最小空闲连接数

min-idle: 15

# 连接池最大阻塞等待时间（使用负值表示没有限制） 默认 -1

max-wait: 30000

<br>

**#自定义配置。expire统一单位为分**

spring:

cache:

multi:

cacheNames: cache1,cache2,cache3

caffeine:

expireAfterWrite: 5000

maximumSize: 1000

redis:

defaultExpiration: 60000

expires:

cache1: 18000

cache2: 18000

cache3: 18000

**### cacheService**

**在service层引用**

1. 注解使用 @Cacheable(value="cacheInfo", key="'cache'+#customer\_id", sync=true)

**### redisService**

```java

@Autowired

private RedisService redisService;

boolean isExist = redisService.setIfPresent(tokenKey.toString(), token, time, TimeUnit.SECONDS);

if(!isExist){

return redisService.set(tokenKey.toString(), token, time, TimeUnit.SECONDS);

}

```

<br>