缓存及 NoSQL 的使用

> **Organization**: 千锋教育 Python 教学部
>
> **Date**: 2019-02-17
>
> **Author**: 张旭

开发任务

1. 为“获取个人资料”接口添加缓存处理
2. 统一为所有数据模型增加缓存处理
    - 任何 model 对象创建时，自动为该对象添加缓存
    - 任何 model 对象修改时，自动更新缓存数据
3. 开发全服人气排行功能
    - 被左滑 -5 分
    - 被右滑 +5 分
    - 被上滑 +7 分
    - 统计全服人气最高的 50 位用户

缓存处理

1. 缓存一般处理流程

```
data = get_from_cache(key)    # 首先从缓存中获取数据
if data is None:
    data = get_from_db()      # 缓存中没有，从数据库获取
    set_to_cache(key, data)   # 将数据添加到缓存，方便下次获取
return data
```

2. Django 的默认缓存接口

```python
from django.core.cache import cache

cache.set('a', 123, 10)
a = cache.get('a')
print(a)
x = cache.incr(a)
print(a)
```

3. Django 中使用 Redis 缓存
   - 安装 django_redis: `pip install django_redis`
   - 配置

```python
# settings 添加如下配置
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/1",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
            "PICKLE_VERSION": -1,
        }
    }
}
```

Redis 的使用

1. Redis 文档
2. 主要数据类型
   - **String 类**: 常用作普通缓存

CMD	Example	Description
set	set('a', 123)	设置值
get	get('a')	获取值
incr	incr('a')	自增
decr	decr('a')	自减
mset	mset(a=123, b=456, c=789)	设置多个值
mget	mget(['a', 'b', 'c'])	获取多个值
setex	setex('kk', 21, 10)	设置值的时候, 同时设置过期时间
setnx	setnx('a', 999)	如果不存在, 则设置该值

- **Hash 类**: 常用作对象存储

CMD	Example	Description
hset	hset('obj', 'name', 'hello')	在哈希表 obj 中添加一个 name = hello 的值
hget	hget('obj', 'name')	获取哈希表 obj 中的值
hmset	hmset('obj', {'a': 1, 'b': 3})	在哈希表中设置多个值
hmget	hmget('obj', ['a', 'b', 'name'])	获取多个哈希表中的值
hgetall	hgetall('obj')	获取多个哈希表中所有的值
hincrby	hincrby('obj', 'count')	将哈希表中的某个值自增 1
hdecrby	hdecrby('obj', 'count')	将哈希表中的某个值自减 1

- **List 类**: 常用作队列(消息队列、任务队列等)

CMD	Example	Description
lpush	lpush(name, *values)	向列表左侧添加多个元素
rpush	rpush(name, *values)	向列表右侧添加多个元素
lpop	lpop(name)	从列表左侧弹出一个元素
rpop	rpop(name)	从列表右侧弹出一个元素
blpop	blpop(keys, timeout=0)	从列表左侧弹出一个元素, 列表为空时阻塞 timeout 秒
brpop	brpop(keys, timeout=0)	从列表右侧弹出一个元素, 列表为空时阻塞 timeout 秒
llen	llen(name)	获取列表长度
ltrim	ltrim(name, start, end)	从 start 到 end 位置截断列表

- **Set 类**: 常用作去重

CMD	Example	Description
sadd	sadd(name, *values)	向集合中添加元素
sdiff	sdiff(keys, *args)	多个集合做差集
sinter	sinter(keys, *args)	多个集合取交集
sunion	sunion(keys, *args)	多个集合取并集
sismember	sismember(name, value)	元素 value 是否是集合 name 中的成员
smembers	smembers(name)	集合 name 中的全部成员
spop	spop(name)	随机弹出一个成员
srem	srem(name, *values)	删除一个或多个成员

- **SortedSet 类**: 常用作排行处理

CMD	Example	Description
zadd	zadd(name, a=12)	添加一个 a, 值为 12
zcount	zcount(name, min, max)	从 min 到 max 的元素个数
zincrby	zincrby(name, key, 1)	key 对应的值自增 1
zrange	zrange(name, 0, -1, withscores=False)	按升序返回排名 0 到最后一位的全部元素
zrevrange	zrevrange(name, 0, -1, withscores=False)	按降序返回排名 0 到最后一位的全部元素
zrem	zrem(name, *value)	删除一个或多个元素

3. 使用 pickle 对 Redis 接口的封装

```python
from pickle import dumps, loads

rds = redis.Redis()

def set(key, value):
    data = dumps(value)
    return rds.set(key, data)

def get(key):
    data = rds.get(key)
    return loads(data)
```

4. 动态修改 Python 属性和方法

```python
class A:
    m = 128
    def __init__(self):
        self.x = 123

    def add(self, n):
        print(self.x + n)

a = A()

# 动态添加属性 (两种方式)
a.y = 456
setattr(a, 'z', 789)

# 动态添加类属性
A.y = 654

# 类属性和实例属性互不影响
print(A.y, a.y)

# 动态添加实例方法
```

```python
def sub(self, n):
    print(self.x - n)
A.sub = sub

# 动态添加类方法
@classmethod
def mul(cls, n):
    print(cls.m * n)
A.mul = mul

# 动态添加静态方法
@staticmethod
def div(x, y):
    print(x / y)
A.div = div

# 属性修改的本质原因
print(A.__dict__, a.__dict__)
```

5. 在 Model 层插入缓存处理
    - Monkey Patch 也叫做 “猴子补丁”, 是一种编程技巧, 旨在运行时为对象动态添加、修改或者替换某项功能