

1. 注册登录接口

- 1.1 jwt 模块的使用
- 1.2 集成到项目中
- 1.3 自定义手机号码校验函数
- 1.4 接口代码实现

2. 获取用户信息接口

- 2.1 请求钩子
- 2.2 装饰器的实现
- 2.3 接口的实现

3. 七牛云介绍

- 3.1 七牛云使用
- 3.2 集成到项目

1. 注册登录接口

1.1 jwt 模块的使用

```
2
3 import jwt
4 from jwt import PyJWTError
5
6 # 加密:
7 # 1. 准备要加密的数据字典
8 # 需要包含 exp 字段, 这个字段的值是 datetime 对象, 是指过期的时间
9 payload = {
10     'user_id': 100,
11     'exp': datetime.now() + timedelta(days=7)
12 }
13 # 2. 准备加密的密钥
14 key = 'N1UzX0FKJRZk5cs1hMDcbqHZ0LKvCAyL85fufewVmUF9bGPA1AXw9w=='
15 # 3. 选择加密的算法 一般是 HS256
16 algorithm = 'HS256'
17 # 4. 生层加密的 token: jwt.encode(要加密的数据字典, key=加密的密钥, algorithm=加密的算法)
18 # 注意: 这里返回的 token 是字节码类型
19 token = jwt.encode(payload, key=key, algorithm=algorithm)
20 print(token)
21
22 # 解密:
23 # 通过 jwt.decode(要解密的 token, key=加密的密钥, algorithm=加密的算法)
24 # 1. 如果 token 合法那么返回数据字典
25 # 2. 如果 token 不合法抛出 jwt.PyJWTError 异常
26 token='xxxx'
27 try:
28     res = jwt.decode(token, key=key, algorithm=algorithm)
29     print(res)
30 except PyJWTError:
```

1.2 集成到项目中

```
1 # 1. 将加密解密的 密钥, 算法, token 有效期配置到项目中
2 # 2. 封装加密过程为单独函数
3 # 3. 封装解密过程为单独函数
```

1.3 自定义手机号码校验函数

```
# 自定义手机号码校验函数
# 1. 定义函数, 接受一个参数
# 2. 函数中校验该参数
#     如果校验失败抛出 ValueError(错误消息)
#     如果校验成功, 返回数据
def parse_mobile(mobile_str):
    if re.match(r'1[3-9]\d{9}', mobile_str):
        return mobile_str
    else:
        raise ValueError('手机号码格式不正确')
```

1.4 接口代码实现

2. 获取用户信息接口

2.1 请求钩子

2.2 装饰器的实现

```

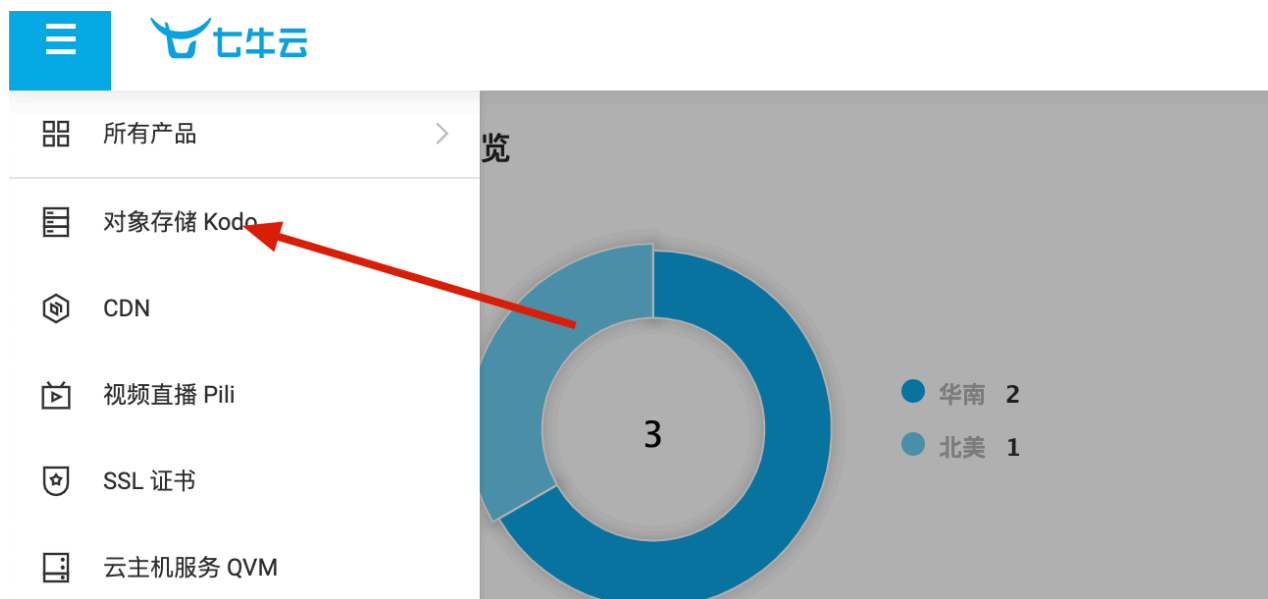
1 from functools import wraps
2
3 from flask import g
4
5
6 def login_required(func):
7     # 1. 判断 g.user_id 是否存在
8     # 1. 不存在 返回 401
9     # 2. 存在调用视图函数
10    @wraps(func)
11    def wrapper(*args, **kwargs):
12        if hasattr(g, 'user_id'):
13            return func(*args, **kwargs)
14        else:
15            return {'message': "用户未登录"}, 401
16
17    return wrapper
18

```

2.3 接口的实现

3. 七牛云介绍

3.1 七牛云使用

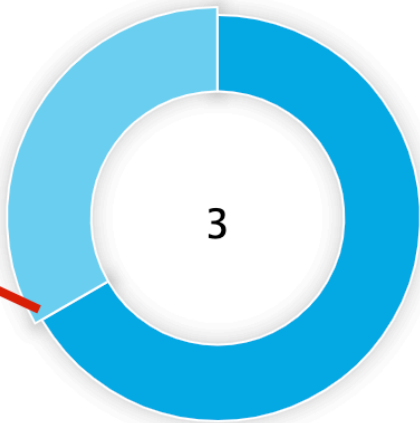


概览

空间管理

跨区域同步

统计分析



● 华南 2
● 北美 1

标准存储

低频存储

归档存储

504.12 KB

7

空间管理

+ 新建空间

刷新列表

全部

华东

华北

华南

北美

东南亚

空间名称

空间类型

存储区域

访问控制

创建时间

py38

自有空间

华南

公开

2021-01-01

hmnews

自有空间

北美

公开

2021-01-01

存储空间名称:

存储空间名称不允许重复，遇到冲突请更换名称。
名称格式为 3 ~ 63 个字符，可以包含小写字母、数字、短划线，且必须以小写字母或者数字开头和结尾。

存储区域: ☐ 华东 ☐ 华北 ☒ 华南 ☐ 北美 ☐ 东南亚

存储空间单价最低。

访问控制: ☒ 公开 ☐ 私有

公开和私有仅对 Bucket 的读文件生效，修改、删除、写入等对 Bucket 的操作均需要拥有者的授权才能进行操作。

存储空间: 华南 存储量: 0 对象数: 0 访问控制: 公开 空间类型: 自有空间

空间概览

文件管理

域名管理

图片样式

转码样式

空间设置

标准存储: ☐ 共 0 个文件 ☐ 共 0 存储量 低频存储: ☐ 共 0 个文件 ☐ 共 0 存储量 归档存储: ☐ 共 0 个文件 ☐ 共 0 存储量

△ 新版浏览器严格限制 http 下载请求，请选用 https 域名或为域名绑定证书

外链域名: 保存为默认域名

上传文件

批量操作

刷新

请输入文件名前缀搜索

<input type="checkbox"/>	文件名	文件类型	存储类型	文件大小	最后更新
--------------------------	-----	------	------	------	------

上传文件至 shpy39

选择上传文件存储类型

标准存储

低频存储

设置路径前缀:

示例: image/**

转码样式:

请选择转码样式

新增转码样式

② 上传覆盖: ☐ 关闭

△上传须知

七牛云存储严禁上传包括反动、及侵权内容的文件。七牛有义务违规文件的用户信息保存, 并保留账号的权利。

返回

选择文件

No file chosen

上传文件

批量操作

刷新

请输入文件名前缀搜索



<input type="checkbox"/>	文件名	文件类型	存储类型	文件大小	最后更新	操作
<input type="checkbox"/>	Xnip2020-12-15_16-42-43.png	image/png	标准存储	203.68 KB	2020-12-15 16:42:46	详情 更多

Xnip2020-12-15_16-42-43.png

基础信息



文件名 Xnip2020-12-15_16-42-43.png

文件类型 image/png

复制文件访问的域名

ETag FkLQ-sx0aUVPw_g9NLnobufQgV8r

文件大小 203.68 KB

存储类型 标准存储

文件链接 http://qldhzpaxz.hn-bkt.clouddn.com/Xnip2020-12-15_16-42-43.png

元数据定义

自定义元数据，在 Header 中返回时都会冠以 x-qn-meta 的前缀，例如，name: value 的元数据，会以 x-qn-meta-name: value 在 Header 中返回。

注意：参数名不区分大小写



772775481@qq.com

个人中心

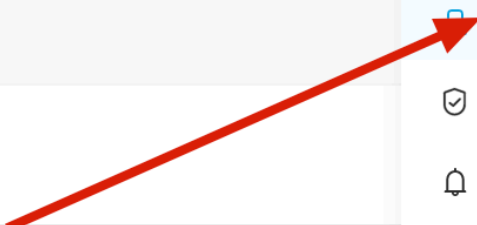
密钥管理

安全设置

提醒设置

操作日志

退出控制台



创建时间

AccessKey/SecretKey

2020-08-24

AK: vtT_LwBjGgt5JGL-bP6im1tldGcZwOGHRNdyWUFp

SK:

显示


```

4 # 定义函数(二进制图片数据)
5 def upload_image(data):
6     # 1. 准备配置信息
7     #     access_key
8     #     secret_key
9     # 需要填写你的 Access Key 和 Secret Key
10    access_key = 'vtT_LwBjGgt5J6L-bP6im1tld6cZw0GHRNdyWUFp'
11    secret_key = 'b5DQrYLidafU-u0oI_NygoTunyZgDHuD7ptFKimf'
12    # 空间名
13    bucket_name = 'shpy39'
14    # 2. 实例化 auth = Auth 对象(access_key, secret_key)
15    auth = Auth(access_key, secret_key)
16    # 3. 生成上传时需要使用的 token = auth.upload_token(空间名)
17    token = auth.upload_token(bucket_name)
18    # 4. 上传图片 put_data(token, key=None, data=图片的二进制数据)
19    ret, resp = put_data(token, key=None, data=data)
20    #     key 是指上传之后, 七牛云存储时用的名字,
21    #     如果是 None 表示七牛云自动计算出文件的唯一名字
22    #     通常情况下, 保持为 None, 让七牛云自动处理
23    # 5. 读取七牛云返回的文件 id
24    #     put_data 返回 一个元组(ret, resp)
25    #     ret['key'] 就是文件的 id
26    # 6. 拼接 域名和文件 id 得到访问图片的完整 url
27    if resp.status_code == 200:
28        key = ret['key']
29        url = 'http://qldhxpaxz.hn-bkt.cloudnn.com/' + key
30        return url
31    else:
32        print('上传失败')
33        print(ret)

```

if __name__ == "__main__": with open('666.png', 'rb') as f

"Datalore" plugin update available
[Update](#) [Plugin Settings...](#) [Ignore this update](#)

3.2 集成到项目

```

22 # 七牛云
23 QINIU_ACCESS_KEY = 'vtT_LwBjGgt5J6L-bP6im1tld6cZw0GHRNdyWUFp'
24 QINIU_SECRET_KEY = 'b5DQrYLidafU-u0oI_NygoTunyZgDHuD7ptFKimf'
25 QINIU_BUCKET_NAME = 'shpy39'
26 QINIU_DOMAIN = 'http://qldhxpaxz.hn-bkt.cloudnn.com/'
27
28

```

```

9      # access_key
10     # secret_key
11     # 需要填写你的 Access Key 和 Secret Key
12     access_key = current_app.config['QINIU_ACCESS_KEY']
13     secret_key = current_app.config['QINIU_SECRET_KEY']
14     # 空间名
15     bucket_name = current_app.config['QINIU_BUCKET_NAME']
16     # 2. 实例化 auth = Auth 对象(access_key, secret_key)
17     auth = Auth(access_key, secret_key)
18     # 3. 生成上传时需要使用的 token = auth.upload_token(空间名)
19     token = auth.upload_token(bucket_name)
20     # 4. 上传图片 put_data(token, key=None, data=图片的二进制数据)
21     ret, resp = put_data(token, key=None, data=data)
22     # key 是指上传之后, 七牛云存储时用的名字,
23     # 如果是 None 表示七牛云自动计算出文件的唯一名字
24     # 通常情况下, 保持为 None, 让七牛云自动处理
25     # 5. 读取七牛云返回的文件 id
26     # put_data 返回 一个元组(ret, resp)
27     # ret['key'] 就是文件的 id
28     # 6. 拼接 域名和文件 id 得到访问图片的完整 url
29     if resp.status_code == 200:
30         key = ret['key']
31         url = current_app.config['QINIU_DOMAIN'] + key
32         return url
33     else:
34         print('上传失败')
35         print(ret)
36         return None
37

```

"Datalore" plugin update available
[Update](#) [Plugin Settings...](#) [Ignore this update](#)

```

9  if __name__ == '__main__':
10     with open('666.png', 'rb') as f:
11         # 判断文件是否为图片类型
12
13         # 方式1: 传递 文件路径/文件对象
14         # type = imghdr.what('666.png')
15         # type = imghdr.what(f)
16         # print(type)
17
18         # 方式2: 传递 二进制数据
19         data_bytes = f.read()
20         type = imghdr.what(None, data_bytes)
21         print(type)

```

```
5
7 # 自定义图片校验函数(参数)
3 def parse_image(image_file):
7     try:
9         file_type = imghdr.what(image_file)
1        except Exception:
2            raise ValueError("文件不是图片")
3
4        if not file_type:
5            raise ValueError("文件不是图片")
6
7        return image_file
3
```