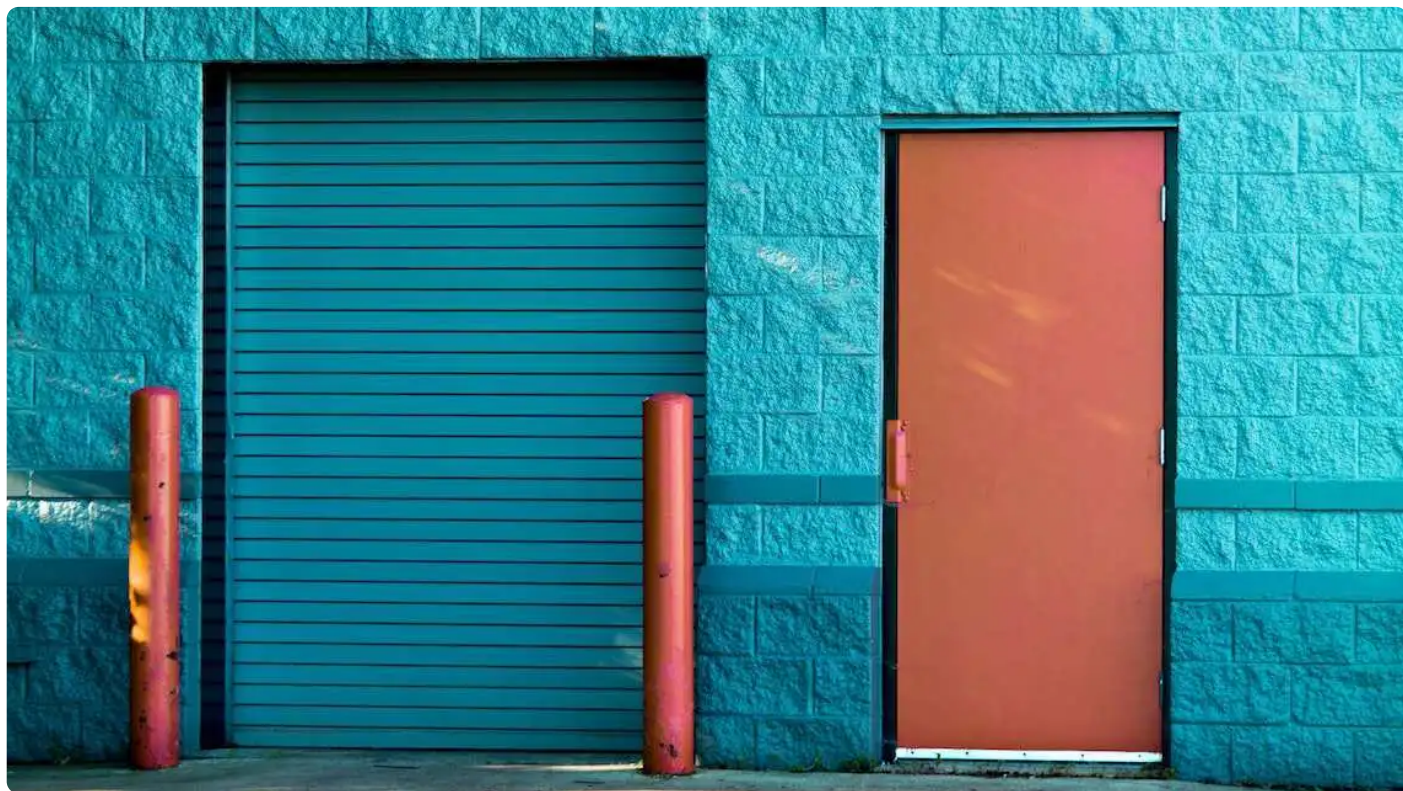


30 | 后端功能接口实战（二）：后端接口该如何开发？

2023-06-30 Barry 来自北京

《Python实战 · 从0到1搭建直播视频平台》



你好，我是 Barry。

在上节课，我们通过用户注册接口的实现，系统学习了如何实现功能接口的开发。这节课，我们继续来学习如何实现视频相关、数据相关的功能接口。

在线视频平台的核心就是视频，而数据模块则是用户以及创作者的“情报部门”，因此这两部分在系统中相当关键。相信有了前面的学习积累，只要你耐心跟着我的思路走，这两部分的接口开发实战你也能轻松跟上。我们先从视频模块的接口分类开始说起。

视频模块接口分类


视频模块整体的功能细分非常全面。不过归纳起来主要就是两个类别。

首先是展示型接口，在视频列表页中，我们要根据不同类别来获取视频数据，在用户点击视频查看详情时，我们需要查询视频相关数据并在前端呈现。其次就是**操作型接口**，比如视频模块相关操作接口的实现，例如点赞、收藏、关注等操作。

接口类型不同，实现思路也有差异，需要我们分类处理。我们先来看看如何实现视频列表的接口。

视频蓝图模块注册


第一步就是模块注册。因为上节课已经完成了系统相关配置，之后的接口开发里我们就不需要单独配置了。这里我们只需要在创建接口 api 包目录的 `__init__.py` 中，完成视频模块的蓝图注册，在原有的代码块下直接写入即可。

 复制代码

```
1 from api.modules.video import video_blu
2     app.register_blueprint(video_blu)
```

创建视频数据库表

完成蓝图的注册后，第二步是实现视频相关的数据库表。我们仍然在项目 `models` 文件下操作，创建 `video.py` 文件，文件中的 `VideoInfo` 类的实现如下所示。

 复制代码

```
1 class VideoInfo(db.Model):
2     __tablename__ = "video_data"
3     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
4     imdb_id = db.Column(db.String(16), unique=True) # IMDB数据库中电影的唯一ID
5     movie_id = db.Column(db.String(16)) # 电影ID
6     tmdb_id = db.Column(db.String(16)) # TMDB数据库中电影的唯一ID
7     title = db.Column(db.String(255)) # 标题
8     little_img_src = db.Column(db.String(255)) # 排行小图片路径 45*67
9     detail_url = db.Column(db.String(255)) # 详情页面url
10    rating = db.Column(db.String(255)) # 排行得分
11    time = db.Column(db.String(255)) # 时长
12    catogery = db.Column(db.String(255)) # 分类
13    show_info = db.Column(db.String(255)) # 上映信息
14    summary_text = db.Column(db.String(255)) # 摘要描述
15    director = db.Column(db.String(255)) # 摘要描述
```

```

16     writers = db.Column(db.String(255)) # 作者
17     stars = db.Column(db.String(255)) # 明星
18     storyline = db.Column(db.String(255)) # 情节描述
19     img_src = db.Column(db.String(255)) # 182-268
20     video_img_src = db.Column(db.String(255)) # 477-268
21     video_href = db.Column(db.String(255)) # 视频详情url
22     true_video_url = db.Column(db.String(1024)) # 真实视频url
23     local_video_src = db.Column(db.String(255)) # 本地视频路径
24     new_video_url = db.Column(db.String(255)) # 服务器视频url
25     local_img_src = db.Column(db.String(255)) # 服务器本地imgurl
26     new_img_url = db.Column(db.String(255)) # 服务器外网imgurl
27     local_video_img_src = db.Column(db.String(255)) # 服务器本地videoimgurl
28     new_video_img_url = db.Column(db.String(255)) # 服务器外网vi
29     download_timeout = db.Column(db.String(16))
30     download_slot = db.Column(db.String(16))
31     download_latency = db.Column(db.String(32))
32     depth = db.Column(db.String(16))

```


以上就是与视频信息相关的表字段。没错，这些字段需要我们在功能开发前，结合需求事先梳理出来的。估计看到这么多字段，你会有点疑惑，别担心，这里你先整体看看有个印象就行，后面的接口开发中，我们用到哪部分字段还会相应去讲解它们的用途。

热度视频接口实现

前面我们说过，接口可以分成展示型接口和操作型接口。

展示型接口的实现逻辑是相似的，无论是推荐视频列表、热门视频列表、电影类别、游戏类别等等，只是视频查询的方式有所不同。我们这就结合热门视频的接口实现代码来看看。

shikey.com转载分享

 复制代码

```

1 @video_blu.route("/hot")
2 @youke_identify
3 def hot():
4     """
5     # 获取热点视频信息
6     # 请求路径: /video/hot
7     # 请求方式: GET
8     :return:
9     """
10    if g.user:
11        user_action_log.warning({
12            'user_id': g.user.id,

```

shikey.com转载分享

```
13         'url': f'/video/hot',
14         'method': 'get',
15         'msg': 'video hot',
16         'event': 'hot',
17     })
18     videos = ContentMain.query.filter_by(audit_status=0).all()
19     choice_videos = random.sample(videos, 8)
20     detail_info = []
21     for i in choice_videos:
22         detail_info.append(i.json())
23     return success(msg='获取热点视频成功', data=detail_info)
24
```

这段代码中，我们要重点关注三个部分。

第一部分是第 10 行代码开始到第 17 行代码，作用是记录日志的行为日志。如果 `g.user` 存在，就表示当前用户是已经登录的状态，其中 `g` 是一个全局对象，用来存储用户信息。`user_action_log` 是一个自定义的日志记录器对象，它在这里充当 `warning` 方法的参数，表示将日志记录为警告级别。其中的参数你可以参考后面的表格。

字段名称	用途
user_id	登录用户的 ID
url	用户请求的 URL 地址
method	请求方法（GET）
msg	请求的原因（在这里是“video hot”，表示用户访问了热门视频页面）
event	事件名称（在这里是“hot”，可以用于后续分析和跟踪）

shikey.com转载分享



`g` 这个全局对象主要用于查询用户状态，我们放在项目的 `api` 文件下的 `utils` 下的 `common.py` 文件内，具体的代码实现如下所示。

```
1 def youke_identify(view_func):
2     @wraps(view_func)
3     def wrapper(*args, **kwargs):
4         response = Auth().identify(request)
5         if response.get('code') == 200:
6             user_id = response.get('data')['user_id']
7             # 查询用户对象
8             user = None
9             if user_id:
10                try:
11                    from api.models.user import UserInfo
12                    user = UserInfo.query.get(user_id)
13                except Exception as e:
14                    current_app.logger.error(e)
15                # 使用g对象保存
16                g.user = user
17
18                return view_func(*args, **kwargs)
19
20         else:
21             g.user = None
22             return view_func(*args, **kwargs)
23
24     return wrapper
25
```

第二块代码我们来看第 18 行，从 ContentMain 表中查询 8 条数据。这里用到了 query 过滤器查询 audit_status 为 0 的所有视频，然后从中随机选择 8 条，存储在 choice_videos。audit_status 就是我们标注的视频状态，表示已经投入平台、可以直接观看的状态。

接下来就是比较核心的第三块代码，也就是代码中的 19 行，这段代码的逻辑是查询完数据后，我们要将数据以 JSON 的格式添加到 detail_info 列表中，最后封装返回 detail_info。到这里，我们就完成了常规数据查询类接口功能的开发。

视频排行榜接口实现

除了常规查询，条件查询接口也很常用。当然这种方式也适用于按视频类别查询。我们以视频排行榜接口为案例，来一起学习实践一下。

我们还是先看一下整体代码，再逐步分析。


```
1 @video_blu.route('/rank')
2 def rank():
3     """
4     获取首页排行榜信息
5     # 请求路径: /video/rank
6     # 请求方式: GET
7     :return: json数据
8         code: 0
9         data:[{"id": id,"title": title},{},...,{"id": id,"title": title}] len=10
10    """
11    rank_video = []
12    # 查询前10条热门
13    try:
14        rank_video = ContentMain.query.filter(ContentMain.status == 0).order_by(C
15            constants.CLICK_RANK_MAX_NEWS).all()
16    except Exception as e:
17        current_app.logger.error(e)
18    if rank_video:
19        # 将视频内容对象列表转成字典列表
20        rank_video_list = []
21        for video in rank_video:
22            rank_video_list.append(video.to_rank_dict())
23
24        data = {
25            "data": rank_video_list,
26        }
27        return success(msg='获取排行榜信息成功', data=data)
28    else:
29        return error(code=HttpCode.db_error, msg='未获取top10视频, 查询有误')
30
```


你会发现整个实现过程中，重点就是数据查询语句该如何设计。

我们需要获取 rank_video 以及查询视频内容表 ContentMain。ContentMain 中有两个条件，第一个是 ContentMain.status == 0，表示可用于平台使用的视频，紧接着是 order_by(ContentMain.clicks.desc())，根据视频点击进行排序，返回数据集即可。

点赞接口的实现

展示型接口的实现我们已经清楚了，接下来我们来看一下操作型接口的实现。这里我们以点赞接口为例，因为它在功能操作接口中具备一定的代表性。你可以先看看代码，再听我分块儿讲

解。

 复制代码

```
1 @video_blu.route('/like/<int:video_id>', methods=['GET', 'POST'])
2 @auth_identify
3 def like(video_id):
4     """
5     点赞
6     :param video_id:
7     :return:
8     """
9     if request.method == 'GET':
10         user_id = g.user.id
11         # 点赞
12         like = ContentLike.query.filter_by(content_id=video_id, user_id=user_id,
13         return success(msg='获取点赞信息成功', data={'like': 1 if like else 0})
14
15 data_dict = request.form
16 user_id = g.user.id
17 like = data_dict.get('like')
18 try:
19     like_event = ContentLike.query.filter_by(content_id=video_id, user_id=user_id)
20 except Exception as e:
21     current_app.logger.error(e)
22     return error(code=HttpCode.db_error, msg='查询点赞出错')
23 if like_event:
24     like_event.status = 1 if like == '1' else 0
25     like_event.update()
26     if like_event.status == 0:
27         user_action_log.warning({
28             'user_id': user_id,
29             'url': f'/video/like/{video_id}',
30             'method': 'post',
31             'msg': 'video cancel like',
32             'event': 'no like',
33         })
34         return success(msg='取消点赞成功')
35     else:
36         user_action_log.warning({
37             'user_id': user_id,
38             'url': f'/video/like/{video_id}',
39             'method': 'post',
40             'msg': 'video like',
41             'event': 'like',
42         })
43         return success(msg='点赞成功')
44 like_event = ContentLike()
```

```

45     like_event.user_id = user_id
46     like_event.content_id = video_id
47     like_event.status = 1 if like == '1' else 0
48     like_event.add(like_event)
49     if like_event.status == 0:
50         user_action_log.warning({
51             'user_id': user_id,
52             'url': f'/video/like/{video_id}',
53             'method': 'post',
54             'msg': 'video cancel like',
55             'event': 'no_like',
56         })
57         return success(msg='取消点赞成功')
58     else:
59         user_action_log.warning({
60             'user_id': user_id,
61             'url': f'/video/like/{video_id}',
62             'method': 'post',
63             'msg': 'video like',
64             'event': 'like',
65         })
66         return success(msg='点赞成功')

```

视频点赞的应用场景就是在用户进入到视频详情页之后，如果对视频内容比较喜欢，可以直接点击点赞按钮来完成点赞的操作。在整个接口实现过程中，我们需要先查询当前用户是否已经点赞过该视频。如果已经点赞，则用户进入界面时，点赞图标就是红色，否则图标颜色为灰色。

我们来梳理一下代码逻辑。第一步是查询状态，代码第 14 行含义是查询 ContentLike 内容表，将结果存储在 like 中，1 表示点赞，0 表示未点赞。

[shikey.com](https://www.shikey.com)转载分享

第二步是信息获取，通过 request.form 获取表单数据，同时获取当前登录用户的 ID，这里直接通过 g.user.id 获取。

[shikey.com](https://www.shikey.com)转载分享


第三步是查询验证。从代码的 19 行开始，我们要查询数据库中是否存在该用户对该视频的点赞记录（即 ContentLike 表中 content_id 和 user_id 与视频 ID 和用户 ID 匹配的记录）。如果查询出现异常，将错误信息记录到日志中，并返回一个错误响应。

从代码第 23 行开始，如果存在点赞记录，我们就根据用户点击的按钮（like 字段）来更新点赞状态，然后更新数据库中的记录。从代码的 47 行开始，如果点赞状态为 0（取消点赞），则记录一个警告日志，并返回一个成功响应，提示用户取消点赞成功。整体实现逻辑就是根据用户操作，更新数据库中的点赞状态，同时要记录相应的用户行为日志。

数据模块接口实战

数据模块中主要涉及与视频相关的数据，我们根据不同维度统计数据并呈现到前端页面。那如何实现数据模块接口？

我们结合查询播放时长前 10 的视频的代码为例一起来看看。

 复制代码

```
1 @video_blu.route('statistics')
2 def statistics():
3     hap = HappyHbase(host='10.20.10.168', port=9090)
4     play_counts_list, play_times_list = hap.scan_start_stop(b'video')
5     hap.close()
6     # 封装数据
7     # 播放时长排行前10
8     time_x_data = [i.get('content_id') for i in play_times_list]
9     time_y_data = [{
10         'value': v.get('play_time'),
11         'name': ContentMain.query.get(v.get('content_id')).first().title,
12         'itemStyle': {
13             'color': color_list[k]
14         }
15     } for k, v in enumerate(play_times_list)]
16     counts_x_data = [i.get('content_id') for i in play_counts_list]
17     counts_y_data = [{
18         'value': v.get('play_counts'),
19         'name': ContentMain.query.get(v.get('content_id')).first().title,
20         'itemStyle': {
21             'color': color_list2[k]
22         }
23     } for k, v in enumerate(play_counts_list)]
24     data = {
25         'time_x_data': time_x_data,
26         'time_y_data': time_y_data,
27         'counts_x_data': counts_x_data,
28         'counts_y_data': counts_y_data,
29     }
30
```

首先，从 HappyHbase 数据库中扫描出视频的播放次数和播放时长的信息。这里调用了 `hap.scan_start_stop(b' video')` 方法，从数据库中扫描出所有视频的播放次数和播放时长的信息，并将结果存储在 `play_counts_list` 和 `play_times_list` 列表中。数据处理之后就会关闭 HappyHbase 连接。

然后，将播放时长和播放次数数据封装成特定的数据结构。

接下来是列表创建环节。我们创建了名为 `time_x_data` 和 `time_y_data` 的列表，分别用来存储播放时长排行前 10 的 `content_id` 和对应的播放时长、标题等信息。再把封装好的数据存储在 `data` 字典中。

最后，接口返回一个成功的响应，状态码为 `ok`，并将封装好的数据作为响应的 `data` 参数返回给客户端。最终前端在系统内呈现数据结果即可。

到这里，我们就完成了用户维度播放量前十视频展示的接口开发，相信通过该接口案例，你实现其他数据模块接口也会非常轻松。

俗话说，磨刀不误砍柴工。实操环节之后，我还帮你整理了接口开发过程中的一些注意事项。希望你能让你在开发过程中少走弯路，全面提升你的接口开发能力，避免因操作不当造成一些不必要的系统错误。

shikey.com转载分享

shikey.com转载分享

开发事件	用途
登录信息安全	用户的登录信息包括手机号码和密码等关键数据。系统要确保所有敏感信息都通过安全的 HTTPS 连接传输，并且在服务器上安全地存储。
异常处理	在处理数据库查询时使用 try / except，这对于处理数据库错误很重要，但也需要确保所有其他可能出现异常的地方都有正确的错误处理。
状态持久化	在处理登录和登出请求时，使用 session 来保存用户的状态。记住，session 中的数据会在用户关闭浏览器或者 session 过期后丢失，所以我们要把临时的、不太重要的数据放在 session 中。
数据更新及时性	在统计用户、视频、评论等数量时，需要确保数据的及时更新。如果数据量很大，则要通过后台任务或者定时任务来定期更新统计信息，而不是在每次请求时都重新计算。
数据保密性	用户的个人数据（如手机号码）在提供给前端之前，必须去除或者替换，以保护用户隐私。



总结

又到了课程的尾声，接下来我们一起回顾总结一下。

这节课，我们聚焦视频模块和数据模块的接口开发。不难发现，课程把重心放在了实现功能逻辑上。我们把接口分为两大类，分别是展示型接口和操作型接口。展示型接口的核心是根据业务需求通过过滤器 query，结合查询条件从而获取数据。在获取到数据之后，一定要对数据进行封装处理，这样更有利于前端数据呈现。

在整个开发过程中我们用到很多的异常处理，希望你注意培养自己的异常处理意识，尽可能考虑周全，在保证程序稳定运行的同时，也要保证用户的优质体验。

shikey.com转载分享

我们选择了点赞操作，作为操作型接口的典型代表。对于操作型接口，我们首先要对操作状态进行查询，在页面加载时，我们就要在前端呈现操作状态。

shikey.com转载分享

对于数据模块的接口，因为我们项目前端主要用 ECharts 来呈现界面，所以要封装处理好返回前端的数据。至于查询过程，我们还是通过对应的需求条件来完成即可。最后，我还整理了一些开发过程中的注意事项，让你能够更全面地掌握接口开发实践。

思考题

我们都知道，用户在点赞和收藏等操作之后，对应操作的视频作品会在“我的关注”列表中展示。那你觉得在“我的关注”列表中是如何实现查询的呢？

欢迎你在留言区和我交流互动，也推荐你把这节课分享给身边更多朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (1)



peter

2023-06-30 来自北京

请教老师两个问题：

Q1：接口开发，有辅助开发的工具吗？

Q2：视频网站主要的成本是带宽吗？现在带宽价格大约是多少？比如1G带宽一个月的费用大致多少？

作者回复：1、我们课程中就是介绍的辅助工具哈，像Flask-RESTful就是，它提供了一些常用的 API 组件，如资源管理器、请求和响应序列化器等，以及一些常用的操作，如过滤、排序、错误处理等。
2、主要成本不是宽带，除了带宽成本，视频网站还需要考虑其他成本，如服务器成本、内容获取成本、版权成本等。这些成本也会对视频网站的运营产生重要的影响。带宽价格也因地区和供应商而异，100-2000的区间，具体你结合地方情况咨询一下。



shikey.com转载分享

shikey.com转载分享