day06 小程序功能开发

内容回顾

。。。

今日概要

- 发布
- 首页
- 详细页面

今日详细

1.发布

1.1 发布流程的问题

- 方式一

```
1. 打开图片进行本地预览
2. 输入文字 & 选择相应的信息
3. 点击发布按钮
    3.1 将本地图片上传到 腾讯云对象存储中COS（oss），并将COS中的图片地址返回。
    3.2 将COS中的图片URL和文字等信息一起提交到后台。
BUG:
    在3.2步骤时可能拿不到COS中的图片。
```

```
function onClickSubmit(){
    // 耗时1分钟，不会阻塞。
    wx.request({
        url:"...",
        success:function(res){
            console.log(res)
        }
    })
    console.log(123);
}
```

- 方式二（推荐）

```
1. 打开图片进行本地预览
2. 将本地图片上传到 腾讯云对象存储中COS
3. 输入文字 & 选择相应的信息
4. 发布：
    必须上传完毕之后，才允许点击发布按钮。
```

1.2 组件：进度条

```
<progress percent="{{percent1}}"  ></progress>

<progress percent="{{percent2}}" activeColor="#DC143C" ></progress>
```

1.3 修改data中的局部数据

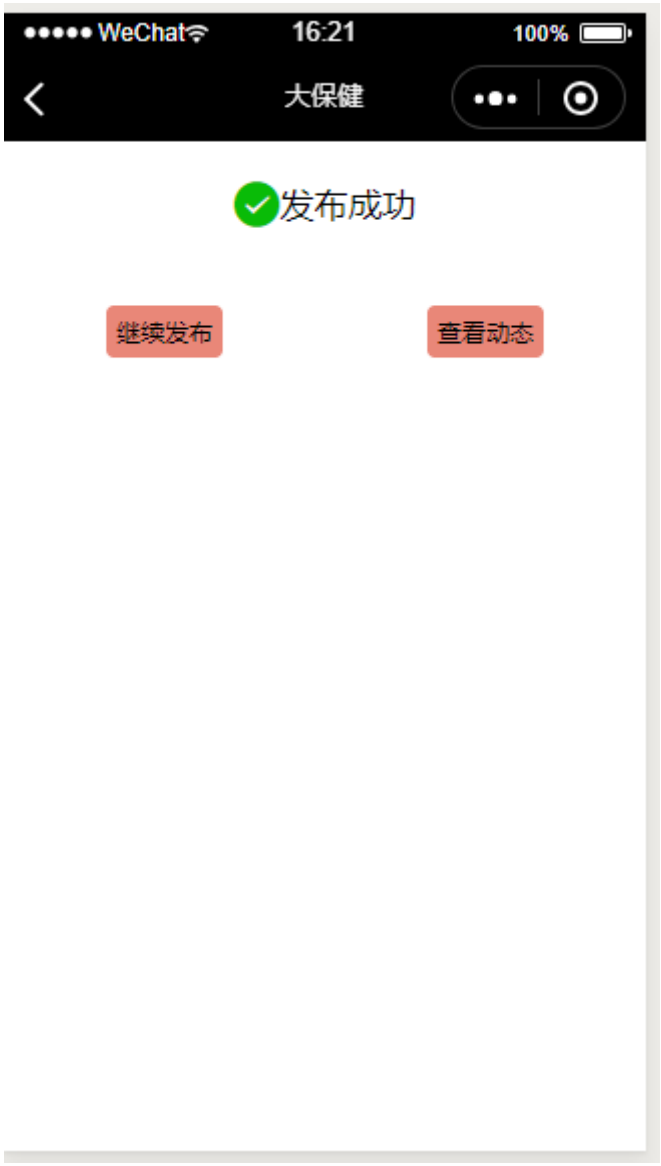
```
<view>-----案例------</view>
<view>点击按钮完成，将图片1的进度条更新为80%</view>
<view wx:for="{{imageList}}">
  <view>{{item.title}}</view>
  <progress percent="{{item.percent}}"  ></progress>
</view>

<button bindtap="changePercent" >点击</button>
```

```
  data: {
    percent1:20,
    percent2:50,
    imageList:[
      {id:1,title:"图片1",percent:20},
      { id: 1, title: "图片2", percent: 30 },
      { id: 1, title: "图片3", percent: 60 },
    ]
  },
  changePercent:function(){
    // 方式1：错误
    /*
    this.setData({
      imageList[0].person: 80
    });
    */

    // 方式2：可以，由于需要全部修改，所以性能差。
    /*
    var dataList = this.data.imageList;
    dataList[0].percent = 80;
    this.setData({
      imageList: dataList
    });
    */

    // 方式3：推荐
    var num = 2;
    this.setData({
      ["imageList[0].percent"]:80,
      ["imageList[" + num + "].percent"]: 90,
      ["imageList[1].title"]:"突突突突突"
    })

  },
```

1.4 发布示例效果

前端


WeChat
16:19
100%
大保健
+图片
来一个
发布
首页
拍卖
发布
消息
我的



后端

- GET 获取临时COS密钥（临时密钥需要有上传和删除的权限）（APIView）

```
config = {
    # 临时密钥有效时长，单位是秒
    'duration_seconds': 1800,
    # 固定密钥 id
    'secret_id': 'AKIDW3Rgszw84ylQxMzNn7KOJ6kFPSL5c5MU',
    # 固定密钥 key
    'secret_key': 'GQSMXmtsjR0QhuIalzTp250nU6digZSD',
    # 换成你的 bucket
    'bucket': 'mini-1251317460',
    # 换成 bucket 所在地区
    'region': 'ap-chengdu',
    # 这里改成允许的路径前缀，可以根据自己网站的用户登录态判断允许上传的具体路径
    # 例子： a.jpg 或者 a/* 或者 * (使用通配符*存在重大安全风险, 请谨慎评估使用)
    'allow_prefix': '*',
    # 密钥的权限列表。简单上传和分片需要以下的权限，其他权限列表请看 https://cloud.tencent.com/document/product/436/31923
    'allow_actions': [
        'name/cos:PostObject',
        'name/cos:DeleteObject',
        # "name/cos:UploadPart",
        # "name/cos:UploadPartCopy",
        # "name/cos:CompleteMultipartUpload",
        # "name/cos:AbortMultipartUpload",
        # "*",
    ],
}
```

  - url

```
url(r'^oss/credential/$', auth.OssCredentialView.as_view()),
```

  - view

```
class OssCredentialView(APIView):

    def get(self, request, *args, **kwargs):
        from utils.tencent.oss import get_credential
        return Response(get_credential())
```

- GET 获取话题接口
  - url

```
url(r'^topic/$', topic.TopicView.as_view()),
```

  - 序列化

```
class TopicSerializer(ModelSerializer):
    class Meta:
        model = models.Topic
        fields = "__all__"
```

  - view（读取所有的话题）
    - APIView
    - ListAPIView（推荐）

```
class TopicView(ListAPIView):
    serializer_class = TopicSerializer
    queryset = models.Topic.objects.all().order_by('-count')
```

- POST 提交 新闻信息
  - url

```
url(r'^news/$', news.NewsView.as_view()),
```

  - 序列化

```
class CreateNewsTopicModelSerializer(serializers.Serializer):
    key = serializers.CharField()
    cos_path = serializers.CharField()


class CreateNewsModelSerializer(serializers.ModelSerializer):
    imageList = CreateNewsTopicModelSerializer(many=True)

    class Meta:
        model = models.News
        exclude = ['user', 'viewer_count', 'comment_count']

    def create(self, validated_data):
        image_list = validated_data.pop('imageList')
        news_object = models.News.objects.create(**validated_data)
        data_list = models.NewsDetail.objects.bulk_create(
            [models.NewsDetail(**info, news=news_object) for info in
image_list]
```

```
        )
        news_object.imageList = data_list

        if news_object.topic:
            news_object.topic.count += 1
            news_object.save()

        return news_object
```

- view（读取所有的话题）
  - APIView
  - CreateAPIView（推荐）

```
class NewsView(CreateAPIView):
    serializer_class = CreateNewsModelSerializer

    def perform_create(self, serializer):
        new_object = serializer.save(user_id=1)
        return new_object
```

1.5 闭包

```
var dataList = ["alex", "changxin", "cck"]
for (var i in dataList) {
  (function(data){
    wx.request({
      url: 'xxxxx',
      success: function (res) {
        console.log(data);
      }
    })
  })(dataList[i])
}
```

2.获取前10条新闻（动态/心情，无需分页）

- url
- view（ListAPIView）
- 序列化

3.复杂版

- 刚进入页面，获取前10条。
  - maxid
  - minid
- 刷新
  - 全局配置

```
{
 "window": {
    "backgroundTextStyle": "dark",
    "navigationBarTitleText": "大保健",
    "enablePullDownRefresh": false
  }
}
```

- 局部配置

```
{
  "usingComponents": {},
  "enablePullDownRefresh": true
}
```

- 停止下拉刷新加载

```
wx.stopPullDownRefresh();
```

- 翻页

4.文章详细页面

任务

1. 表结构的设计
2. 发布页面，将数据存储到数据库。
    - 图片
    - 文本
    - 话题
    - 地址
    - 后端：
        - APIView
        - ListAPIView
        - ListAPIView
    - 主要事项: 用户登录不涉及（用户ID先写死了）
3. 简化版：首页

    - 仅需要前10条数据
    - 不需要：刷新和翻页
4. 简化版：详细页面
    - 展示详细信息
        - 轮播图
        - 新闻信息
        - 在页面上展示所有的一级评论
5. 复杂版：首页
    - 下拉
    - 上翻
6. 复杂版：详细页面
    - 添加一条评论
    - 多级评论（不用做）