**开发者文档**

## **技术栈概览**

### **前端**

1. **框架**：微信小程序（WXML, WXSS, JavaScript）
2. **UI组件**：微信小程序内置组件
3. **网络请求**：wx.request API
4. **文件处理**：wx.downloadFile, wx.openDocument API
5. **状态管理**：Page数据绑定 (this.setData)

### **后端**

1. **框架**：Flask (Python)
2. **数据库**：MySQL
3. **对象存储**：腾讯云COS (Cloud Object Storage)
4. **服务部署**：PM2进程管理
5. **API设计**：RESTful风格

## **开发流程概述**

1. **需求分析与设计**
2. 资源列表浏览、筛选
3. 资源下载
4. 资源收藏管理
5. 用户身份标识
6. **数据库设计**
7. 资源表 (exam\_resources)
8. 用户收藏表 (user\_favorites)
9. 下载录表 (user\_downloads)
10. **API开发**
11. 资源列表API
12. 下载API
13. 收藏API
14. **前端开发**
15. 页面布局
16. 资源列表展示
17. 筛选功能
18. 下载功能
19. 收藏功能
20. **调试与修复**
21. API联调
22. Bug修复
23. 性能优化

### **1. 数据结构设计**

#### 资源模型

{

  id: Number,           // 资源ID

  title: String,        // 资源标题

  description: String,  // 资源描述

  fileType: String,     // 文件类型

  fileSize: Number,     // 文件大小

  downloadCount: Number,// 下载次数

  year: Number,         // 年份

  category: String,     // 分类

  examType: String,     // 考试类型

  fileId: String,       // 对应COS的文件ID

  isFavorite: Boolean   // 是否收藏

}

**主要数据表**

**exam\_resources: 存储所有考试资源信息**

**user\_favorites: 存储用户收藏关系**

**user\_downloads: 记录下载历史**

### **2. 后端API**

#### 2.1 资源列表AP

GET /api/exam/resources

参数:

- exam\_type (必须): 考试类型

- category (可选): 分类

- year (可选): 年份

- user\_id (可选): 用户ID，用于返回是否收藏

#### 2.2 收藏列表API

POST /api/exam/favorite

参数:

- resource\_id (必须): 资源ID

- user\_id (必须): 用户ID

- action (必须): add/remove

#### 2.4 下载API

POST /api/exam/download

参数:

- resource\_id (必须): 资源ID

- user\_id (可选): 用户ID，用于记录下载历史

### **3. 前端关键函数实现**

#### 3.1 资源加载

async loadResources() {

  // 1. 设置loading状态

  this.setData({ loading: true });

  // 2. 构建查询参数

  const params = {

    exam\_type: this.data.activeType

  };

  if (this.data.activeYear) params.year = this.data.activeYear;

  if (this.data.activeCategory) params.category = this.data.activeCategory;

  if (this.data.userId) params.user\_id = this.data.userId;

  // 3. 转换为查询字符串

  const queryString = Object.keys(params)

    .map(key => `${encodeURIComponent(key)}=${encodeURIComponent(params[key])}`)

    .join('&');

  // 4. 发送API请求

  return new Promise((resolve, reject) => {

    wx.request({

      url: `${this.data.apiBaseUrl}/api/exam/resources?${queryString}`,

      method: 'GET',

      header: { 'Content-Type': 'application/json', 'Accept': 'application/json' },

      success: (response) => {

        // 处理响应...

        if (response.statusCode === 200 && response.data && response.data.code === 0) {

          const responseData = response.data.data || [];

          // 格式化资源数据

          const resources = responseData.map(item => ({

            id: item.id,

            title: item.title || '未命名资源',

            description: item.description || '暂无描述',

            fileType: item.file\_type || 'pdf',

            fileSize: this.formatFileSize(item.file\_size || 0),

            downloadCount: item.download\_count || 0,

            year: item.year || 2024,

            category: item.category || '',

            examType: item.exam\_type || this.data.activeType,

            fileId: item.file\_id || '',

            isFavorite: item.is\_favorite || false

          }));

          // 更新状态

          this.setData({

            resources,

            filteredResources: resources,

            loading: false

          });

          resolve(resources);

        } else {

          // 错误处理...

          reject(new Error(response.data?.message || '获取资源失败'));

        }

      },

      fail: (err) => {

        // 网络错误处理...

        reject(new Error('网络请求失败'));

      }

    });

  });

}

3.2 文件下载功能

handleDownload(e) {

  // 1. 获取资源ID和资源信息

  const resourceId = e.currentTarget.dataset.id;

  const resource = this.data.resources.find(r => r.id === resourceId);

  if (!resource) {

    wx.showToast({ title: '资源不存在', icon: 'error' });

    return;

  }

  try {

    // 2. 显示加载状态

    wx.showLoading({ title: '准备下载...' });

    // 3. 构建请求数据

    const requestData = {

      resource\_id: resourceId,

      user\_id: this.data.userId || 'anonymous'

    };

    // 4. 请求下载链接

    wx.request({

      url: `${this.data.apiBaseUrl}/api/exam/download`,

      method: 'POST',

      data: requestData,

      header: {

        'Content-Type': 'application/json',

        'Accept': 'application/json'

      },

      success: (response) => {

        // 5. 处理响应

        if (response.statusCode !== 200 || response.data.code !== 0 || !response.data.data || !response.data.data.download\_url) {

          console.error('API返回错误数据:', response.data);

          wx.hideLoading();

          wx.showToast({ title: '获取下载链接失败', icon: 'error' });

          return;

        }

        // 6. 提取下载信息

        const downloadUrl = response.data.data.download\_url;

        const fileName = response.data.data.file\_name;

        const fileType = response.data.data.file\_type;

        // 7. 下载文件

        wx.showLoading({ title: '下载中...' });

        const downloadTask = wx.downloadFile({

          url: downloadUrl,

          success: (res) => {

            if (res.statusCode === 200) {

              wx.hideLoading();

              // 8. 打开文件预览

              wx.openDocument({

                filePath: res.tempFilePath,

                fileType: fileType.toLowerCase(),

                showMenu: true,

                success: () => {

                  // 9. 更新下载次数(UI)

                  const resources = [...this.data.resources];

                  const index = resources.findIndex(r => r.id === resourceId);

                  if (index !== -1) {

                    resources[index].downloadCount += 1;

                    this.setData({

                      resources,

                      filteredResources: this.filterResourcesByCurrentCriteria(resources)

                    });

                  }

                },

                fail: (err) => {

                  console.error('打开文件失败:', err);

                  wx.showToast({ title: '打开失败', icon: 'error' });

                }

              });

            } else {

              wx.hideLoading();

              wx.showToast({ title: '下载文件失败', icon: 'error' });

            }

          },

          fail: (err) => {

            console.error('下载失败:', err);

            wx.hideLoading();

            wx.showToast({ title: '下载失败', icon: 'error' });

          }

        });

        // 10. 添加下载进度监听

        downloadTask.onProgressUpdate((res) => {

          wx.showLoading({

            title: `下载中 ${res.progress}%`,

          });

        });

      },

      fail: (err) => {

        console.error('API请求失败:', err);

        wx.hideLoading();

        wx.showToast({ title: '请求失败', icon: 'error' });

      }

    });

  } catch (error) {

    console.error('处理失败:', error);

    wx.hideLoading();

    wx.showToast({ title: '加载失败', icon: 'error' });

  }

}

#### 3.3 收藏功能

async toggleFavorite(e) {

  // 1. 验证用户登录状态

  if (!this.data.userId) {

    wx.showToast({ title: '请先登录', icon: 'none' });

    return;

  }

  // 2. 获取资源ID和收藏状态

  const resourceId = e.currentTarget.dataset.id;

  const isFavorite = this.data.isFavorite[resourceId];

  try {

    // 3. 构建请求数据

    const requestData = {

      resource\_id: resourceId,

      user\_id: this.data.userId,

      action: isFavorite ? 'remove' : 'add'

    };

    // 4. 发送收藏请求

    const response = await wx.request({

      url: `${this.data.apiBaseUrl}/api/exam/favorite`,

      method: 'POST',

      data: requestData

    });

    // 5. 处理响应

    if (response.statusCode !== 200 || response.data.code !== 0) {

      throw new Error(response.data?.message || '操作失败');

    }

    // 6. 更新本地收藏状态

    const newIsFavorite = { ...this.data.isFavorite };

    if (isFavorite) {

      delete newIsFavorite[resourceId];

    } else {

      newIsFavorite[resourceId] = true;

    }

    this.setData({ isFavorite: newIsFavorite });

    // 7. 显示操作结果

    wx.showToast({

      title: isFavorite ? '已取消收藏' : '已收藏',

      icon: 'success'

    });

  } catch (error) {

    console.error('收藏操作失败:', error);

    wx.showToast({ title: '操作失败', icon: 'error' });

  }

}

### **4. 后端关键功能实现**

#### 4.1 COS客户端集成

class COSClient:

    def \_\_init\_\_(self):

        self.client = None

    def get\_client(self):

        if self.client is None:

            try:

                config = CosConfig(

                    Region=COS\_CONFIG["region"],

                    SecretId=COS\_CONFIG["secret\_id"],

                    SecretKey=COS\_CONFIG["secret\_key"]

                )

                self.client = CosS3Client(config)

                logger.info("COS client initialized")

            except Exception as e:

                logger.error(f"COS client error: {str(e)}")

                raise e

        return self.client

    def get\_file\_list(self, prefix=""):

        """获取存储桶中的文件列表"""

        client = self.get\_client()

        try:

            response = client.list\_objects(

                Bucket=COS\_CONFIG["bucket"],

                Prefix=prefix

            )

            files = []

            if "Contents" in response:

                for item in response["Contents"]:

                    files.append({

                        "key": item["Key"],

                        "last\_modified": item["LastModified"],

                        "size": item["Size"],

                        "etag": item["ETag"]

                    })

            logger.info(f"Retrieved {len(files)} files from COS")

            return files

        except Exception as e:

            logger.error(f"Error getting file list: {str(e)}")

            raise e

    def get\_presigned\_url(self, key, expires=3600):

        """获取预签名临时下载URL"""

        client = self.get\_client()

        try:

            url = client.get\_presigned\_url(

                Method='GET',

                Bucket=COS\_CONFIG["bucket"],

                Key=key,

                Expired=expires

            )

            logger.info(f"Generated presigned URL for {key}")

            return url

        except Exception as e:

            logger.error(f"Error generating presigned URL: {str(e)}")

            raise e

#### 4.2 下载API实现

@exam\_bp.route("/api/exam/download", methods=["POST"])

def get\_download\_url():

    """获取文件下载链接"""

    try:

        data = request.json

        resource\_id = data.get("resource\_id")

        user\_id = data.get("user\_id", "anonymous")

        # 验证参数

        if not resource\_id:

            return jsonify({

                "code": 400,

                "message": "Missing resource\_id",

                "data": None

            }), 400

        # 获取资源信息

        resource = db.execute\_query("SELECT \* FROM exam\_resources WHERE id = %s", [resource\_id])

        if not resource:

            return jsonify({

                "code": 404,

                "message": "Resource not found",

                "data": None

            }), 404

        resource = resource[0]

        # 获取临时下载链接

        download\_url = cos\_client.get\_presigned\_url(resource["file\_id"], expires=3600)

        # 记录下载

        if user\_id != "anonymous":

            db.execute\_insert(

                "INSERT INTO user\_downloads (user\_id, resource\_id, ip\_address) VALUES (%s, %s, %s)",

                [user\_id, resource\_id, request.remote\_addr]

            )

        # 更新下载次数

        db.execute\_update(

            "UPDATE exam\_resources SET download\_count = download\_count + 1 WHERE id = %s",

            [resource\_id]

        )

        logger.info(f"Generated download URL for resource\_id={resource\_id}, user\_id={user\_id}")

        return jsonify({

            "code": 0,

            "message": "success",

            "data": {

                "download\_url": download\_url,

                "file\_name": resource["title"],

                "file\_type": resource["file\_type"]

            }

        })

    except Exception as e:

        logger.error(f"Error generating download URL: {str(e)}")

        return jsonify({

            "code": 500,

            "message": f"Error: {str(e)}",

            "data": None

        }), 500

#### 4.3 收藏API实现

@exam\_bp.route("/api/exam/favorite", methods=["POST"])

def toggle\_favorite():

    """切换收藏状态"""

    try:

        data = request.json

        resource\_id = data.get("resource\_id")

        user\_id = data.get("user\_id")

        action = data.get("action")  # 'add' or 'remove'

        # 验证参数

        if not resource\_id or not user\_id or action not in ["add", "remove"]:

            return jsonify({

                "code": 400,

                "message": "Invalid parameters",

                "data": None

            }), 400

        # 检查资源是否存在

        resource = db.execute\_query("SELECT \* FROM exam\_resources WHERE id = %s", [resource\_id])

        if not resource:

            return jsonify({

                "code": 404,

                "message": "Resource not found",

                "data": None

            }), 404

        # 根据action执行操作

        if action == "add":

            # 先检查是否已存在

            existing = db.execute\_query(

                "SELECT \* FROM user\_favorites WHERE user\_id = %s AND resource\_id = %s",

                [user\_id, resource\_id]

            )

            if not existing:

                db.execute\_insert(

                    "INSERT INTO user\_favorites (user\_id, resource\_id) VALUES (%s, %s)",

                    [user\_id, resource\_id]

                )

        elif action == "remove":

            db.execute\_update(

                "DELETE FROM user\_favorites WHERE user\_id = %s AND resource\_id = %s",

                [user\_id, resource\_id]

            )

        logger.info(f"Toggle favorite: action={action}, resource\_id={resource\_id}, user\_id={user\_id}")

        return jsonify({

            "code": 0,

            "message": "success",

            "data": {

                "action": action,

                "resource\_id": resource\_id,

                "user\_id": user\_id

            }

        })

    except Exception as e:

        logger.error(f"Error toggling favorite: {str(e)}")

        return jsonify({

            "code": 500,

            "message": f"Error: {str(e)}",

            "data": None

        }), 500

## **调试与修复记录**

### **关键问题与解决方案**

#### 腾讯云COS客户端配置问题

**问题**：使用错误的参数名expires导致生成预签名URL失败**解决方案**：

1. 查看腾讯云COS SDK文档，确认get\_presigned\_url方法的正确参数是Expired（大写E）
2. 修改get\_presigned\_url方法，使用Method='GET'和Expired参数，与腾讯云SDK匹配
3. 重启服务器后重新运行服务，确保修改生效

#### 微信小程序API请求问题

**问题**：使用async/await方式调用wx.request，但无法正确获取结果**解决方案**：

1. 将wx.request改为回调方式，确保正确处理响应
2. 增加更详细的错误处理和日志记录
3. 添加请求头Content-Type和Accept以确保正确的数据格式

#### 数据库连接问题

**问题**：MySQL连接数据包序列号错误**解决方案**：

1. 优化数据库连接池管理
2. 处理MySQL连接异常，包括自动重连机制

#### 前端UI优化

**问题**：下载进度展示不直观**解决方案**：

1. 使用downloadTask.onProgressUpdate监听下载进度
2. 使用wx.showLoading展示动态下载进度

## **部署与维护**

### **部署步骤**

1. **后端部署**
2. 上传代码到服务器
3. 安装依赖：Flask, PyMySQL, 腾讯云SDK等
4. 配置PM2启动脚本：pm2 start app.py --name exam-api --interpreter python3
5. **前端部署**
6. 在微信开发者工具中构建小程序
7. 上传代码至微信小程序平台
8. 提交审核并发布

### **维护注意事项**

1. **定期备份**
2. 数据库定期备份
3. COS资源定期复制
4. **监控与日志**
5. 使用PM2监控后端服务状态
6. 保留详细日志，包括API请求、错误等
7. 定期检查服务器资源使用情况
8. **性能优化**
9. 定期清理不再使用的资源
10. 优化数据库查询
11. 监控并优化API响应时间

## **总结**

本项目是一个完整的微信小程序+后端API的解决方案，专注于考试资料的分享和管理。前端使用微信小程序技术栈，后端使用Flask框架和MySQL数据库，文件存储采用腾讯云COS。关键功能包括资源浏览、筛选、下载和收藏，用户体验良好。开发过程中遇到的主要技术挑战包括腾讯云COS集成、微信小程序API调用和数据库连接管理等，这些问题都通过仔细调试和参考文档得到了解决。

“注意:在开发过程中遇到了一个区分大小写的错误”

ERROR:utils.cos\_client:Error generating presigned

URL: get\_presigned\_download\_url() got an unexpected keyword argument 'expires'

问题在于

在controllers/exam\_resources.py中，  
调用COS客户端时使用的参数是expires（小写），  
但修改后的COS客户端函数接受的参数是Expired（大写E）