2025年夏季《移动软件开发》实验报告

姓名：陈怡冰 学号：23020007007

姓名和学号?	**陈怡冰，23020007007**
本实验属于哪门课程?	中国海洋大学25夏《移动软件开发》
实验名称?	实验4：媒体API之口述校史
博客地址?	OUC-2025年夏季《移动软件开发》实验报告-实验四-CSDN博客
Github仓库地址?	ChenFirstIce/Mobile-software-development-for-the-summer-semester-of-2025: OUC-2025夏季学期《移动软件开发》课程的六个实验以及个人项目的课程报告以及代码。主要是使用微信小程序开发工具开发微信小程序，并涉及到一点鸿蒙系统。

（备注：将实验报告发布在博客、代码公开至 github 是 **加分项**，不是必须做的）

一、实验目标

1、掌握视频API的操作方法；

2、掌握如何发送随机颜色的弹幕。

二、实验步骤

2.1 准备工作

1. **准备视频地址**
    - 实验文档中已经提供四个视频地址
    - 也可以自行爬虫获得所有的视频地址
        - 使用python进行爬取，并导出json形式，然后加入到`pages/index/index.js`的属性`list`中

```
from bs4 import BeautifulSoup
import re
import urllib.request
import json
import urllib.parse
import time


findlink = re.compile(r'<a.*?href="(.*?)".*?>(.*?)</a>')
findvideo = re.compile(r'vurl="([^"]*)"')
findtitle = re.compile(r'<title>(.*?)</title>')
```

```
def main():
    baseurl = "https://arch.ahnu.edu.cn/ksxs/ksxsdyq.htm"
    datalist = getdata(baseurl)
    savepath = "video.json"
    saveData(datalist,savepath)


def getdata(baseurl):
    datalist = []
    head = {}
    head['User-Agent'] = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124
Safari/537.36'

    # 首先获取主页面
    request = urllib.request.Request(baseurl,headers=head)
    response = urllib.request.urlopen(request)
    html = response.read().decode('utf-8')
    soup = BeautifulSoup(html,'html.parser')

    for link in soup.find_all('div', class_='text'):
        item = re.findall(findlink,str(link))[0]
        url = item[0]
        title = re.findall('span class="">(.*?)</span>',str(item))
[0]

        #对url进行处理
        if url.startswith('../'):
            url = urllib.parse.urljoin(baseurl, url)
        else:
            url = baseurl + url
        print(f"正在爬取: {title} - {url}")

        detail_data = get_detail_data(url, head)
        if detail_data:
            detail_data['id'] = len(datalist) + 1
            detail_data['title'] = title
            datalist.append(detail_data)
        time.sleep(1)
    return datalist


def get_detail_data(url, headers):
    try:
        request = urllib.request.Request(url, headers=headers)
        response = urllib.request.urlopen(request)
        html = response.read().decode('utf-8')
        soup = BeautifulSoup(html, 'html.parser')

        video_links = []

        #<script name="_videourl" vurl="..."></script>
        for script in soup.find_all('script'):
            script_text = str(script)
            # 查找vurl="..."格式
```

```
            video_matches = findvideo.findall(script_text)
            for video_url in video_matches:
                if video_url.startswith('/'):
                    full_video_url = urllib.parse.urljoin(url,
video_url)
                else:
                    full_video_url = video_url
                video_links.append(full_video_url)
                print(f"找到视频链接(vurl): {full_video_url}")


        # 去重
        video_links = list(set(video_links))

        print(f"页面 {url} 共找到 {len(video_links)} 个视频链接")

        return {
            'url': url,
            'video_links': video_links,
        }

    except Exception as e:
        print(f"爬取链接 {url} 时出错: {e}")
        return None


def saveData(datalist, savepath):
    json_data = []
    for data in datalist:
        json_item = {
            'id': data['id'],
            'title': data['title'],
            'videourl': data['video_links']  # 将video_links重命名为
video_urls
        }
        json_data.append(json_item)

    # 写入JSON文件
    with open(savepath, 'w', encoding='utf-8') as f:
        json.dump(json_data, f, ensure_ascii=False, indent=2)

    print(f"数据已保存到 {savepath}")
    print(f"共保存了 {len(json_data)} 条记录")


if __name__ == "__main__":
    main()
```

- 将爬取集成到微信小程序中

```
  /*创建视频上下文，控制视频的播放和停止 */
  onLoad: function (options) {
    this.videoCtx = wx.createVideoContext('myVideo')
```

```
    this.startCrawling()
  },

  /*播放视频 */
  playVideo: function(e){
    this.videoCtx.stop()
    this.setData({
     src: e.currentTarget.dataset.url
    } )
  },

  getDanmu: function(e){
    ……
  },

  sendDanmu: function(e){
    ……
  },


  startCrawling: function() {
    this.setData({
      isLoading: true,
      crawlStatus: '开始爬取视频数据...'
    })

    this.crawlVideoData()
  },

  crawlVideoData: function() {
    const baseUrl = "https://arch.ahnu.edu.cn/ksxs/ksxsdyq.htm"

    console.log('开始请求主页面:', baseUrl)

    wx.request({
      url: baseUrl,
      method: 'GET',
      header: {
        'Content-Type': 'text/html; charset=utf-8'
      },
      success: (res) => {
        console.log('主页面请求成功，状态码:', res.statusCode)
        if (res.statusCode === 200) {
          console.log('HTML内容长度:', res.data.length)
          this.parseMainPage(res.data)
        } else {
          this.setData({
            isLoading: false,
            crawlStatus: `网络请求失败，状态码: ${res.statusCode}`
          })
        }
      },
      fail: (err) => {
        console.error('请求失败:', err)
        this.setData({
          isLoading: false,
```

```
          crawlStatus: '网络请求失败: ' + err.errMsg
        })
      }
    })
  },

  parseMainPage: function(html) {
    try {
      const simpleLinkRegex = /<a[^>]*href="([^"]*)"[^>]*>/g
      const allLinks = []
      let match

      while ((match = simpleLinkRegex.exec(html)) !== null) {
        allLinks.push(match[1])
      }

      console.log('找到的所有链接:', allLinks.slice(0, 10)) // 只显示前
10个

      const textDivRegex = /<div[^>]*class="text"[^>]*>(.*?)
<\/div>/gs
      const linkRegex = /<a[^>]*href="([^"]*)"[^>]*>(.*?)<\/a>/g
      const titleRegex = /<span[^>]*class=""[^>]*>(.*?)<\/span>/g

      const videoLinks = []
      let textDivMatch

      while ((textDivMatch = textDivRegex.exec(html)) !== null) {
        const divContent = textDivMatch[1]
        console.log('找到text div:', divContent.substring(0, 200))

        let linkMatch

        // 在每个div中查找链接
        while ((linkMatch = linkRegex.exec(divContent)) !== null) {
          const url = linkMatch[1]
          const linkText = linkMatch[2]

          console.log('找到链接:', url, '文本:', linkText)

          // 提取标题
          const titleMatch = titleRegex.exec(linkText)
          if (titleMatch) {
            const title = titleMatch[1].trim()
            let fullUrl = url

            // 处理URL
            if (url.startsWith('../')) {
              fullUrl = 'https://arch.ahnu.edu.cn/' +
url.substring(3)
            } else if (!url.startsWith('http')) {
              fullUrl = 'https://arch.ahnu.edu.cn/ksxs/' + url
            }

            console.log(`找到视频链接: ${title} -> ${fullUrl}`)
```

```
            videoLinks.push({
              title: title,
              url: fullUrl
            })
          }
        }
      }

      console.log(`总共找到 ${videoLinks.length} 个视频页面`)

      this.setData({
        crawlStatus: `找到 ${videoLinks.length} 个视频页面，开始获取视频链接...`
      })

      // 逐个获取视频详情
      this.crawlVideoDetails(videoLinks, 0)

    } catch (error) {
      console.error('解析主页面失败:', error)
      this.setData({
        isLoading: false,
        crawlStatus: '解析页面失败: ' + error.message
      })
    }
  },

  // 爬取视频详情页面
  crawlVideoDetails: function(videoLinks, index) {
    if (index >= videoLinks.length) {
      // 所有视频都爬取完成
      this.setData({
        isLoading: false,
        crawlStatus: '爬取完成'
      })
      return
    }

    const videoLink = videoLinks[index]

    wx.request({
      url: videoLink.url,
      method: 'GET',
      header: {
        'Content-Type': 'text/html; charset=utf-8'
      },
      success: (res) => {
        if (res.statusCode === 200) {
          const videoUrls = this.extractVideoUrls(res.data, videoLink.url)

          if (videoUrls.length > 0) {
            const newVideo = {
              id: this.data.list.length + 1,
              title: videoLink.title,
              videourl: videoUrls
```

```
            }

            this.setData({
              list: [...this.data.list, newVideo],
              crawlStatus: `正在爬取: ${videoLink.title} (${index +
1}/${videoLinks.length})`
            })
          }
        }

        setTimeout(() => {
          this.crawlVideoDetails(videoLinks, index + 1)
        }, 1000)//timespeed(1)
      },
      fail: (err) => {
        console.error(`爬取 ${videoLink.title} 失败:`, err)
        console.error(`失败的URL: ${videoLink.url}`)
        // 即使失败也继续下一个
        setTimeout(() => {
          this.crawlVideoDetails(videoLinks, index + 1)
        }, 1000)
      }
    })
  },

  // 从页面中提取视频URL
  extractVideoUrls: function(html, baseUrl) {
    const videoUrls = []

    // 查找 vurl="..." 格式的视频链接
    const vurlRegex = /vurl="([^"]*)"/g
    let match

    while ((match = vurlRegex.exec(html)) !== null) {
      let videoUrl = match[1]

      // 处理URL
      if (videoUrl.startsWith('/')) {
        videoUrl = 'https://arch.ahnu.edu.cn' + videoUrl
      } else if (!videoUrl.startsWith('http')) {
        videoUrl = baseUrl + videoUrl
      }

      videoUrls.push(videoUrl)
    }

    // 去重
    return [...new Set(videoUrls)]
  },
```

2. **一个播放的图片play.png，实验文档已经提供，也可以自行在icon网站上下载**

2.2 创建小程序

1. **视图设计**

   - 视图设计

     区域划分为三个部分：

     - 视频播放

       `controls`属性用于显示暂停/播放，需要设置`video`是`enable-danmu danmu-btn`，否则没有弹幕无法在视频中显示：

       ```
       <video id="myVideo" src="{{src}}" controls enable-danmu danmu-btn>
       </video>
       ```

     - 弹幕区域

       横向分成两个小区域，分别是**输入区**以及**发送区域。**

       ```
       <view class='danmuArea'>
         <input type='text' placeholder='请输入弹幕内容' value =
       '{{danmuTxt}}' bindinput='getDanmu'></input>
         <button bindtap='sendDanmu'>发送弹幕</button>
       </view>
       ```

     - 视频列表

       通过`wx:for`控制循环，`wx:key="video{{index}}"`自动遍历。

       ```
       <view class='videoList'>
         <view class='videoListTitle'>视频列表 ({{list.length}} 个)</view>
         <view class='videoBar' wx:for="{{list}}" wx:key="video{{index}}"
       data-url="{{item.videourl}}" bindtap='playVideo'>
           <image src='/images/play.png'></image>
           <text>{{item.title}}</text>
         </view>
       </view>
       ```

- 样式设计

  全部来自实验文档，重点是学会使用`flex-grow`以及`margin`来控制样式。

  ```
  /* 视频播放器样式 */
  video{
    width: 100%;
  }

  /* 视频列表样式 */
  ```

```
.videoList{
  width: 100%;
  min-height: 400rpx;
}

.videoListTitle {
  font-size: 36rpx;
  font-weight: bold;
  color: #987838;
  text-align: center;
  padding: 20rpx;
  background-color: #f0f0f0;
  border-bottom: 2rpx solid #987838;
}

.videoBar{
  width: 95%;
  display: flex;
  flex-direction: row;
  border-bottom: 1rpx solid #987838;
  margin: 10rpx;
  background-color: #ffffff;
  border-radius: 8rpx;
  box-shadow: 0 2rpx 4rpx rgba(0,0,0,0.1);
}

.videoBar:active {
  background-color: #f0f0f0;
}

text{
  font-size: 45rpx;
  color:  #987838;
  margin: 20rpx;
  flex-grow: 1;
}

image{
  width: 70rpx;
  height: 70rpx;
  margin: 20rpx;
}

/* 弹幕控制区域样式 */
.danmuArea{
  display: flex;
  flex-direction: row;
}

input{
  border: 1rpx solid #987838;
  flex-grow: 1;
  height: 100rpx;
  font-size: 40rpx;
}
```

```
button{
  color: white;
  background-color: #987838;
  border-radius: 8rpx;
}
```

- 效果如下：

2. 视频播放逻辑实现

- 设置data属性

```
data: {
    list: []
}
```

- 创建视频上下文`videoCtx`，控制视频的播放和停止。

```
  onLoad: function (options) {
    this.videoCtx = wx.createVideoContext('myVideo')
    this.startCrawling()
  },
```

- 获得视频链接，播放视频

```
  /*播放视频 */
  playVideo: function(e){
    this.videoCtx.stop()
    this.setData({
     src: e.currentTarget.dataset.url
    } )
  },
```

3. 弹幕逻辑实现

- 设置data属性

```
data: {
    danmuTxt: '',
    list: []
}
```

- 获得前端`input`的弹幕内容

```
  getDanmu: function(e){
    this.setData({
      danmuTxt: e.detail.value
    })
  },
```

- 发送弹幕

`function getRandColor()`：通过随机0~256之间的数并转换成16进制以此获得随机RGB数，使得弹幕的颜色随机变化。

使用微信自带的弹幕功能发送弹幕。

```
sendDanmu: function(e){
  function getRandColor() {
    let rgb = []
    for(let i = 0;i < 3;i++){
      let color = Math.floor(Math.random()*256).toString(16)
      color = color.length == 1 ? '0' + color:color
      rgb.push(color)
    }
    return '#' + rgb.join('')
  }

  let text = this.data.danmuTxt;
  this.videoCtx.sendDanmu({
    text:text,
    color: getRandColor()
  })
},
```

三、程序运行结果

- 直接使用json导入视频链接的界面
- 集成爬取视频链接

四、问题总结与体会

通过本次实验，我深入掌握了微信小程序开发的核心技术。在循环结构方面，学会了使用wx:for指令实现视频列表的动态渲染，以及JavaScript中的while循环进行正则表达式匹配和数据处理。在视频导入方面，掌握了`<video>`组件的使用方法，包括视频播放控制、弹幕功能实现，以及通过wx.createVideoContext创建视频上下文进行程序化控制。同时，通过编写Python爬虫脚本，学会了使用BeautifulSoup解析HTML、正则表达式提取数据、以及将爬取结果保存为JSON格式，为小程序提供数据源。这次实验让我深刻体会到前端开发与数据获取技术的结合应用，提升了解决实际问题的能力。