

一、实验准备

课程主页: [课程主页\(gitee.com\)](https://gitee.com)

实验文档: [lab2.pdf\(gitee.com\)](https://gitee.com)

学习视频: [第二个小程序 \(1\) bilibili.com](https://bilibili.com)

二、实验目标

- 1、掌握服务器域名配置和临时服务器部署;
- 2、掌握 wx.request 接口的用法。

三、实验步骤

1、“和风天气”API密钥申请

和风天气官方网址为[和风天气\(gweather.com\)](https://gweather.com)。

和风天气开发服务提供了API、iOS SDK和Android SDK用以访问基于位置的天气数据, 包括实况天气、30天预报、逐小时预报、空气质量AQI, 灾害预警、分钟级降水、生活指数等天气数据服务。

当我们选择“免费用户”类型, 使用邮箱进行注册并激活后可以获取三天之内全球各地区的实时天气, 支持的免费接口调用流量基本上可以满足我们这次试验的开发学习要求。

密钥申请步骤如下:

(1) 进入开发服务控制台

(2) 进入项目管理

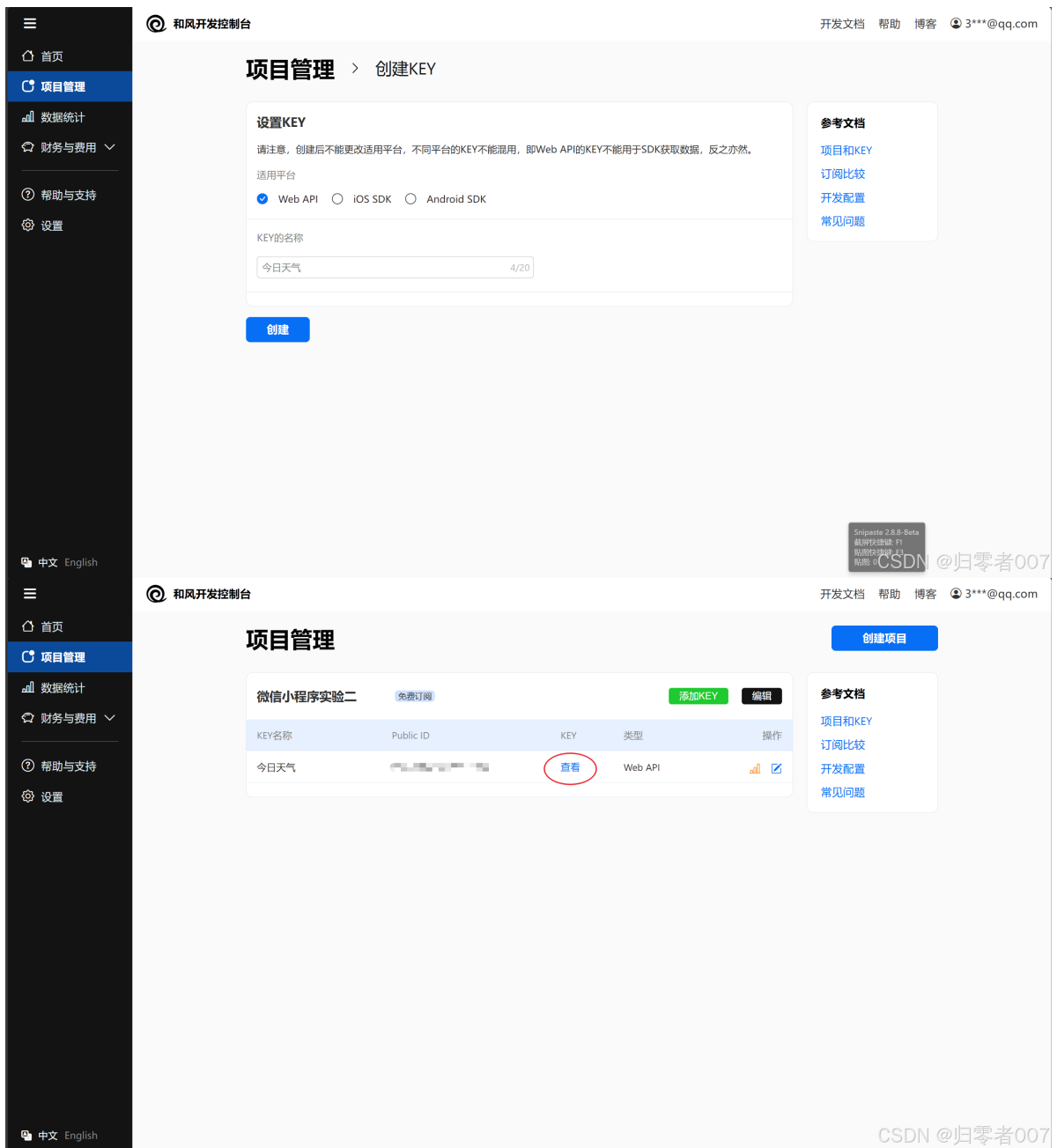
(3) 创建项目

- 填写项目名称
- 选择“免费订阅”
- 设置KEY, 适用平台为“Web API”, 并填写KEY的名称
- 点击“创建”

(4) 添加KEY

- 设置KEY, 适用平台为“Web API”, 并填写KEY的名称
- 点击“创建”

(5)查看密钥并保存



2、服务器域名配置

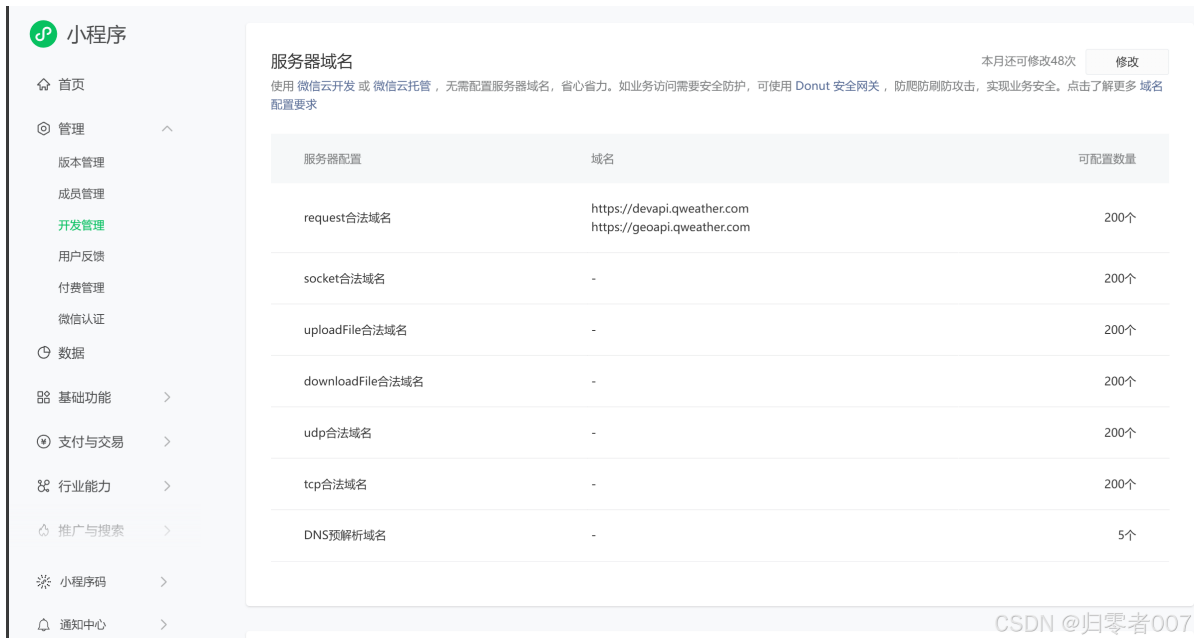
当我们从小程序中读取天气有关的信息时，需要访问“和风天气”的服务器，因此需要对相关域名地址进行服务器配置。所需的域名地址如下：

<https://devapi.qweather.com>

<https://geoapi.qweather.com>

服务器域名配置步骤如下：

- 进入[微信公众平台](#)
- 登陆，进入“管理”
- 进入“开发管理”
- 进入“开发设置”
- 下拉到服务器域名，点击修改，将上述两个接口添加到“request合法域名”中。结果如下图：



3、项目配置

(1) 创建新项目

类似实验一，这里不再赘述。

(2) 导入和风天气图标素材

- 项目新建images文件夹
- 进入[和风天气图标\(qweather.com\)](https://qweather.com)，下载图标素材压缩包，并解压到项目images文件夹

4. 视图设计

4.1 导航栏设计

在 app.json 文件中自定义导航栏标题和背景颜色。

```
{
  "pages": [
    "pages/index/index",
    "pages/logs/logs"
  ],
  "window": {
    "navigationBarTextStyle": "white",
    "navigationBarTitleText": "实验二：天气查询小程序",
    "navigationBarBackgroundColor": "#3C5F81"
  },
  "style": "v2",
  "componentFramework": "glass-easel",
  "sitemapLocation": "sitemap.json",
  "lazyCodeLoading": "requiredComponents"
}
```

#####

4.2 页面设计

(1) 页面结构

index.wxml代码如下:

```
<!--index.wxml-->
<view class='container' >
  <!-- 区域1: 地区选择器 -->
  <picker mode='region' bindchange="changeRegion">
    <view>{{region}}</view>
  </picker>

  <!-- 区域2: 文本区域 -->
  <text>{{now.temp}} {{now.text}}</text>

  <!-- 区域3: 图片区域 -->
  <image src='/images/Qweather-Icons-1.6.0/icons/{{now.icon || 999}}.svg'></image>

  <!-- 区域4: 多行天气信息 -->
  <view class='detail'>
    <view class='bar'>
      <view class='box'>湿度</view>
      <view class='box'>气压</view>
      <view class='box'>能见度</view>
    </view>
    <view class='bar'>
      <view class='box'>{{now.humidity}}%</view>
      <view class='box'>{{now.pressure}}hPa</view>
      <view class='box'>{{now.vis}}Km</view>
    </view>
    <view class='bar'>
      <view class='box'>风向</view>
      <view class='box'>风速</view>
      <view class='box'>风力</view>
    </view>
    <view class='bar'>
      <view class='box'>{{now.windDir}}</view>
      <view class='box'>{{now.windSpeed}} Km/h</view>
      <view class='box'>{{now.windScale}} 级</view>
    </view>
  </view>
</view>
```

(2) 页面样式

index.wxss代码如下:

```
/**index.wxss**/
page {
  height: 100vh;
  display: flex;
  flex-direction: column;
}
```

```

.container{
  height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-around;
}

.detail{
  width: 100%;
  display: flex;
  flex-direction: column;
}

.bar{
  display: flex;
  flex-direction: row;
  margin: 20rpx 0;
}

.box{
  width: 33%;
  text-align: center;
}

text{
  font-size: 80rpx;
  color: #3C5F81;
}

```

5. 逻辑实现

5.1 主要数据：

- region: ['山东省', '青岛市', '黄岛区'], region以数组的形式表示地区信息，默认为黄岛
- location: '山东省青岛市黄岛区', location以字符串的形式表示地理位置信息
- id: 101120206, id表示城市的Location ID
- now: "", now表示当前天气信息，默认为空

5.2 主要逻辑：

- getLocationId 函数：获得查询地址的唯一的Location ID，并将其保存在id中
- getWeather 函数：调用getLocationId 函数，对获得的 Location ID获取实时的天气数据，并将其保存在now中
- changeRegion 函数：一个 picker 地址选择器，以三级列表的形式筛选想要查询的城市。当城市地址变化时，调用getWeather 函数。

index.js文件代码如下：

```

// pages/index/index.js
Page({
  data: {

```

```

    region: ['山东省', '青岛市', '黄岛区'], // 默认地区
    location: '山东省青岛市黄岛区', // 地理位置
    id: 101120206, // 城市ID
    now: '' // 当前天气信息
  },

  // 当地区改变时调用
  changeRegion: function(e) {
    this.setData({
      region: e.detail.value // 更新地区数据
    });
    this.getweather();
  },

  // 获取城市Location ID
  getLocationId: function() {
    let that = this;
    wx.request({
      url: 'https://geoapi.qweather.com/v2/city/lookup',
      data: {
        location: that.data.region, // 使用当前地区数据
        key: 'Your_key' //使用你保存的key
      },

      success(res) {
        console.log(res.data.location[0].id)
        that.setData({
          id: res.data.location[0].id // 设置城市ID
        });
      }
    });
  },

  // 获取当前天气信息
  getweather: function() {
    let that = this;
    that.getLocationId();
    wx.request({
      url: 'https://devapi.qweather.com/v7/weather/now',
      data: {
        location: that.data.id, // 使用获取到的城市Location ID
        key: 'Your_key'
      },

      success(res) {
        console.log(res.data.now)
        that.setData({
          now: res.data.now // 设置当前天气信息
        });
      }
    });
  },

  // 页面加载时调用
  onLoad(options) {
    this.getweather();
  },

```

```
onReady() {}, // 页面初次渲染完成时调用

onShow() {}, // 页面显示时调用

onHide() {}, // 页面隐藏时调用

onUnload() {}, // 页面卸载时调用

onPullDownRefresh() {}, // 用户下拉动作时调用

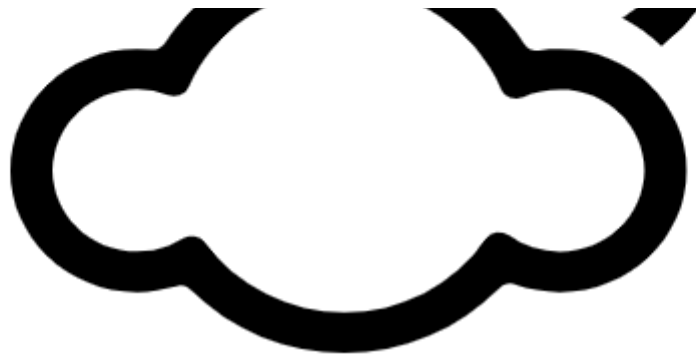
onReachBottom() {}, // 页面上拉触底事件的处理函数

onShareAppMessage() {} // 用户点击右上角分享时调用
});
```

四、程序运行结果

程序的最终运行结果如下：





湿度

97%

气压

1004hPa

能见度

16Km

风向

东北风

风速


19 Km/h

风力

3 级

CSDN @归零者607

19:15

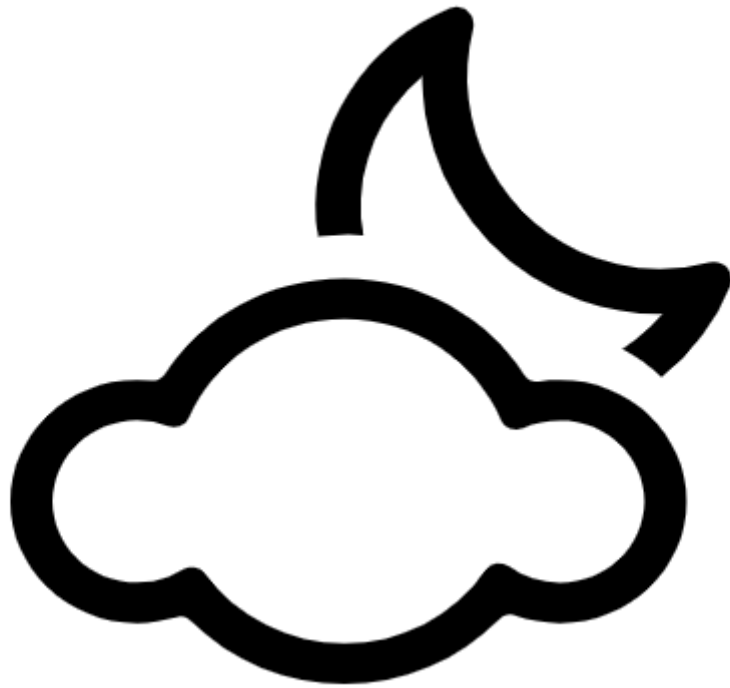
89% 

实验二：天气查询小程序



天津市,天津市,和平区

27 多云



湿度	气压	能见度
80%	1004hPa	30Km
风向	风速	风力
东风	12 Km/h	3 级

五、问题总结与体会

问题：wx.request方法调用异常

1. 问题描述：

wx.request方法使用实验文档提供的请求URL和两个查询参数，返回错误代码 400，说明请求错误，可能包含错误的请求参数或缺少必选的请求参数。

2. 解决方法：

通过查阅实时天气API使用方法后发现，该请求需要两个必选的请求参数，分别为key 和 location，其中 key 需要传递的参数是之前保存的密钥，与实验文档的描述相一致；location 需要传递的参数是和风天气查询地区的唯一标识Location ID，与实验文档的描述不同。

阅读开发文档后，发现问题可以通过城市搜索API[城市搜索 for API](#) | [和风天气开发服务 \(qweather.com\)](#)获得，因此在JS文件中加入了getLocationId 函数，用来获得查询城市的locationID，并将其作为参数传递到getWeather函数中。

实时天气

编辑

平台: [API](#) [iOS](#) [Android](#)

获取中国3000+市区和海外20万个城市实时天气数据，包括实时温度、体感温度、风力风向、相对湿度、大气压强、降水量、能见度、露点温度、云量等。

注意：实况数据均为近实时数据，相比真实的物理世界有5-20分钟的延迟，请根据实况数据中的 `obsTime` 确定数据对应的准确时间。

请求URL

```
GET https://api.qweather.com/v7/weather/now?{查询参数}
```

如果是免费订阅，将上述API Host更改为 `devapi.qweather.com`。参考[免费订阅可用的数据](#)。

请求参数

请求参数包括必选和可选参数，参数之间使用 `&` 进行分隔。

- `key` (**必选**) 用户认证key，请参考[如何获取你的KEY](#)。支持[数字签名](#)方式进行认证。例如 `key=123456789ABC`
- `location` (**必选**) 需要查询地区的`LocationID`或以英文逗号分隔的[经度,纬度坐标](#)（十进制，最多支持小数点后两位），`LocationID`可通过[GeoAPI](#)获取。例如 `location=101010100` 或 `location=116.41,39.92`
- `lang` 多语言设置，请阅读[多语言](#)文档，了解我们的多语言是如何工作、如何设置以及数据是否支持多语言。
- `unit` 数据单位设置，可选值包括 `unit=m`（公制单位，默认）和 `unit=i`（英制单位）。更多选项和说明参考[度量衡单位](#)。

CSDN @归零者007

城市搜索

编辑

平台: [API](#) [iOS](#) [Android](#)

城市搜索API提供全球地理位置、全球城市搜索服务，支持经纬度坐标反查、多语言、模糊搜索等功能。

天气数据是基于地理位置的数据，因此获取天气之前需要知道具体的位置信息。使用城市搜索，可获取到该城市的基本信息，包括城市的Location ID（你需要这个ID去查询天气），多语言名称、经纬度、时区、海拔、Rank值、归属上级行政区域、所在行政区域等。

另外，城市搜索也可以帮助你在你的APP中实现模糊搜索，用户只需要输入1-2个字即可获得结果。

请求URL

```
GET https://geoapi.qweather.com/v2/city/lookup?{查询参数}
```

CSDN @归零者007

请求参数

请求参数包括必选和可选参数，如不填写可选参数将使用其默认值，参数之间使用 `&` 进行分隔。

- `location` (**必选**) 需要查询地区的名称，支持文字、以英文逗号分隔的[经度,纬度坐标](#)（十进制，最多支持小数点后两位）、`LocationID`或`Adcode`（仅限中国城市）。例如 `location=北京` 或 `location=116.41,39.92`

模糊搜索，当location传递的为文字时，支持模糊搜索，即用户可以只输入城市名称一部分进行搜索，最少一个汉字或2个字符，结果将按照相关性和[Rank值](#)进行排列，便于开发或用户进行选择他们需要查看哪个城市的天气。例如 `location=bei`，将返回与bei相关性最强的若干结果，包括黎巴嫩的贝鲁特和中国的北京市

重名，当location传递的为文字时，可能会出现重名的城市，例如陕西省西安市、吉林省辽源市下辖的西安区和黑龙江省牡丹江市下辖的西安区，此时会根据[Rank值](#)排序返回所有结果。在这种情况下，可以通过 `adm` 参数的方式进一步确定需要查询的城市或地区，例如 `location=西安&adm=黑龙江`

- `key` (**必选**) 用户认证key，请参考[如何获取你的KEY](#)。支持[数字签名](#)方式进行认证。例如 `key=123456789ABC`
- `adm` 城市的上级行政区划，可设定只在某个行政区划范围内进行搜索，用于排除重名城市或对结果进行过滤。例如 `adm=beijing`

如请求参数为 `location=chaoyang&adm=beijing` 时, 返回的结果只包括北京市的朝阳区, 而不包括辽宁省的朝阳市
如请求参数仅为 `location=chaoyang` 时, 返回的结果包括北京市的朝阳区、辽宁省的朝阳市以及长春市的朝阳区

- `range` 搜索范围, 可设定只在某个国家或地区范围内进行搜索, 国家和地区名称需使用ISO 3166 所定义的国家代码。如果不设置此参数, 搜索范围将在所有城市。例如 `range=cn`
- `number` 返回结果的数量, 取值范围1-20, 默认返回10个结果。
- `lang` 多语言设置, 请阅读[多语言](#)文档, 了解我们的多语言是如何工作、如何设置以及数据是否支持多语言;DN @归零者007

- getLocationId方法如下:

```
// 获取城市ID
getLocationId: function() {
  let that = this;
  return new Promise((resolve, reject) => {
    wx.request({
      url: 'https://geoapi.qweather.com/v2/city/lookup',
      data: {
        location: that.data.region, // 使用当前地区数据
        key: 'Your_key'
      },
    },
    success(res) {
      if (res.data && res.data.location && res.data.location[0]) {
        that.setData({
          id: res.data.location[0].id // 设置城市ID
        });
        resolve(); // 成功后调用resolve
      } else {
        reject('Location ID not found'); // 如果未找到城市ID, 调用reject
      }
    },
    fail(err) {
      reject(err); // 请求失败时调用reject
    }
  });
});
```

3. 收获与体会:

在遇到wx.request方法调用异常的问题之后, 我从官网找到正确的 API 接口, 在使用过程中, 与实验文档给出的内容对比, 发现传入参数的错误, 从而想到仿照已有的范例函数getWeather, 编写类似的函数 getLocationId获得Location ID作为中转, 再将Location ID返回到函数 getWeather 中获取实时天气数据, 从而完成实验二的任务。

在这个过程中, 我提高了分析问题、解决问题的能力, 顺利完成了实验目标, 即掌握服务器域名配置和临时服务器部署, 以及掌握 wx.request 接口的用法。

