第八章在小程序里查询天气（下）

在上一节，我们已经简单地完成一个天气查询小程序的静态页面，现在，我们就来为它加上逻辑代码。

获取当前位置天气

想象一下，当小程序加载，默认进入天气详情页面，此时页面初始化时应该做哪些工作呢？

初始化逻辑分 3 步走，具体如下：

1. 首先获取当前地理位置信息，取得经纬度。
2. 调用腾讯地理位置逆解析服务，获取当前位置信息。
3. 调用心知天气 API，获取当前城市天气。

首先我们需要获取当前地理位置信息。小程序提供了 wx.getLocation API，可以让我们有能力获取到当前使用者的地理位置信息。

wx.getLocation 使用方法如下：

```
wx.getLocation({
      success: function (res) { // success 属性是获取成功回调函数
        console.log(res)
        that.setData({
          hasLocation: true,
          location: formatLocation(res.longitude, res.latitude)
        })
      }
    })
```

我解释一下上述语句：我们给 wx.getLocation 函数传递了一个对象，对象中有 success 属性，该属性为一个 Function 类型，用于绑定获取地理位置成功事件的回调，在这个回调事件中，我们可以拿到一个 res 对象，res 对象包含 longitude 和 latitude，分别是经度、纬度。

接下来我们要通过经纬度获取城市信息，这里我们使用地理位置逆解析服务，使用前需要在腾讯地图注册，申请 API，然后请求 http://apis.map.qq.com/ws/geocoder/v1/?location=，取得当前城市信息。

发起请求的代码如下：

```
wx.request({
      url:  'http://apis.map.qq.com/ws/geocoder/v1/',
      data:  {
        location:  '39.984154,116.307490'
      },
      success:  function(result)  {
        //  ...
      },
      fail:  function({errMsg})  {
        //  ...
      }
    })
```

我们使用 wx.request API，这个 API 让我们拥有发起网络请求的能力，这样我们就可以通过访问腾讯地图提供的 restful API 来解析经纬度数据了。

下面是参数说明：

- url：小程序请求 URL
- data ：query 参数，location 为经纬度信息
- success：成功回调
- fail：失败回调

在成功回调中我们可以，接下来就可以根据城市来获取天气信息了，调用心知天气 API 前同样需要在心知天气注册并申请 API KEY。

我们依然使用 wx.request API

```
 wx.request({
     url:  'https://api.seniverse.com/v3/weather/now.json',
```

```
    data: {
      key: 'API_KEY',
      location: '北京'
    },
    success: function(result) {
      // ...
    },
    fail: function({errMsg}) {
      // ...
    }
  }}
```

接下来我们继续开发地图页面，我们使用 map 组件，这个组件可以在页面上创建一个可交互的地图。

根据地图获取天气信息

页面结构搭建完毕后，我们开始编写页面逻辑。 这个页面的重点在于初始化 map 组件，主要有两步：

1. 创建 map 组件并显示。
2. 在 map 组件上创建每个城市的标记点。

在 demo 中提供了全国省会城市的经纬度信息，在地图页面 目录下，名称为 loc.js，城市经纬度信息结构如下：

```
{
    name:'广州市',  //  城市名称
    latlng:'113.264385,23.129112'  //  对应的城市经纬度信息
}
```

然后我们在 map.js 引入该文件，将其转换成 marker 数组，用于在页面上显示标记：

```
const locData = require('./loc.js')
var markers = locData.map(loc=>{
    let latlng = loc.latlng.split(',') // 经纬度信息原本是以字符
```

```
串形式保存，这里要将其分割成 2 部分
    return  {
        id:  loc.name,
        latitude:  parseFloat(latlng[1]),  //  经纬度信息需要转换成数
值型
        longitude:  parseFloat(latlng[0]),//  经纬度信息需要转换成数
值型
        name:  loc.name,
        iconPath:'/image/location.png',
    }
})
```

当用户点击 marker，我们可以用 wx.switchTab API 导航到天气详情页，并带上城市信息，这里由于 switchTab API 的限制，无法在 URL 中传递参数，因此我们在全局对象 app 上设置 currentCity 属性来保存当前选中城市。

```
var  app  =  getApp()
        app.globalData.currentCity  =  e.markerId
        wx.switchTab({
            url:  '/page/index/index',
        })
```

我们回到天气详情页面，我们需要在 index.js 做下修改，当页面载入时，判断是否有指定城市，如果有则获取指定城市的天气信息

```
  onShow:  function(option){
          this.reloadData()
    }
```

map.js 完整代码如下：

```
const  locData  =  require('./loc.js')
var  markers  =  locData.map(loc=>{
    let  latlng  =  loc.latlng.split(',')
    return  {
        id:  loc.name,
        latitude:  parseFloat(latlng[1]),
        longitude:  parseFloat(latlng[0]),
        name:  loc.name,
        iconPath:'/image/location.png',
```

```
    }
})
Page({
    data:  {
        latitude:  markers[0].latitude,
        longitude:  markers[0].longitude,
        markers:  markers,
    },
    switchCity(e){
        //console.log(e)
        var  app  =  getApp()
        app.globalData.currentCity  =  e.markerId
        wx.switchTab({
            url:  '/page/index/index',
        })
    }
})
```

至此，我们就完成了一个天气查询的小程序了。