关键点设计

由 李东杰创建, 最后修改于八月 01, 2018

目的

经过对小程序的调研与验证，证实小程序实现目前用户端的主要交易流程可行。但也存在一些难点，主要如下：

- **异步处理**

  小程序提供的框架使用回调的方式处理异步，而在我们的交易流程中存在大量的异步操作，这些异步操作又交织着异常处理流程，如果全用回调的方式处理，很大可能会陷入回调地狱，后期维护性很差。

- **蓝牙连接/交互的管理**

  蓝牙连接和交互跟交易紧密相关，正是上文中提及异步处理和异常流程的焦点。微信默认提供的蓝牙接口只有adapter一层，这一层拥有对蓝牙的开启/扫描/连接/读写的所有功能。如果直接使用adapter进行连接，势必在异常处理中不停地对蓝牙状态进行判断。小程序的文档中提到，对于蓝牙的开关/扫描开始与结束等指令，最好结对调用(如多次开启一次结束可能会引发异常)。如何对蓝牙连接和交互进行管理，以在不同交易阶段和异常分支中提供简单清晰的状态和统一的操作，也是难点之一。

- **开发环境**

  在微信提供的默认环境中进行开发的主要问题有 语法检测功能弱/代码编辑管理效率低等，需要配置新的开发环境。

方案

经过调研与验证，采用如下方案解决以上问题:

- **主要采用 ES2017(draft) 中 async/await 语法进行异步处理。**

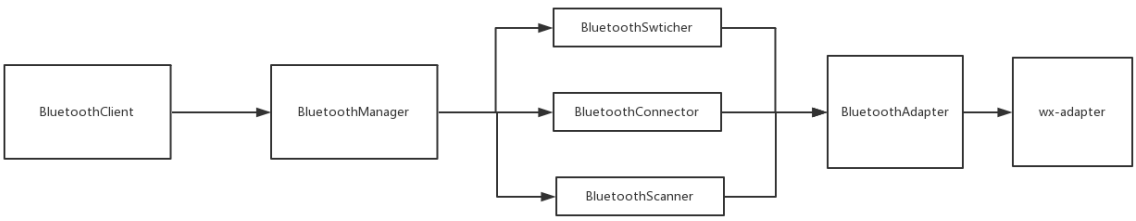
  async/await是基于Promise的异步处理语法，其最大的优势是以同步风格的代码编写异步代码，这样大大方便了异步代码的组织，可阅读性较好。但小程序原生目前只支持到ES6(ES2015)的部分语法。

  经过调研，选择了FACEBOOK官方的 async/await 运行时库，只要在使用到async/await语法的文件中引入该库，即可正常使用。该库的原理是在当前上下文的全局对象上补充了async/await对应的功能函数，将对应语法的函数转变为 ES5 语法的代码。以其原理分析，该库可以保证语法的兼容性在微信框架内得到支持。

  示例:

```
connectDevice: async function() {
  wait(50)
  const result = await BluetoothClient.connectDevice(this.data.macAddressFromServer)
  if (result.success) {
    console.log(currentPath, "已完成连接设备操作，并更改了设备的notify为true", result)
    this.handleDeviceMessage()
    this.shakeHands()
  } else {
    console.log(currentPath, "连接到设备出错", result)
    this.setData({
      pageStatus: PAGE_STATUS.CONNECT_FAIL
    })
  }
},
```

- **对蓝牙连接进行分层管理。将蓝牙状态的管理封装如下管理器**



  - BluetoothAdapter 包装微信api，主要是提供promise化的接口
  - BluetoothSwitcher 管理蓝牙的开关状态及操作
  - BluetoothScanner 管理蓝牙扫描的状态与操作
  - BluetoothConnector 管理蓝牙连接的状态与操作
  - BluetoothManager 管理上述BluetoothSwitcher/BluetoothScanner/BluetoothConnector。
  - BluetoothClient 提供直接面向使用页面的接口。更高程度地封装相关服务。

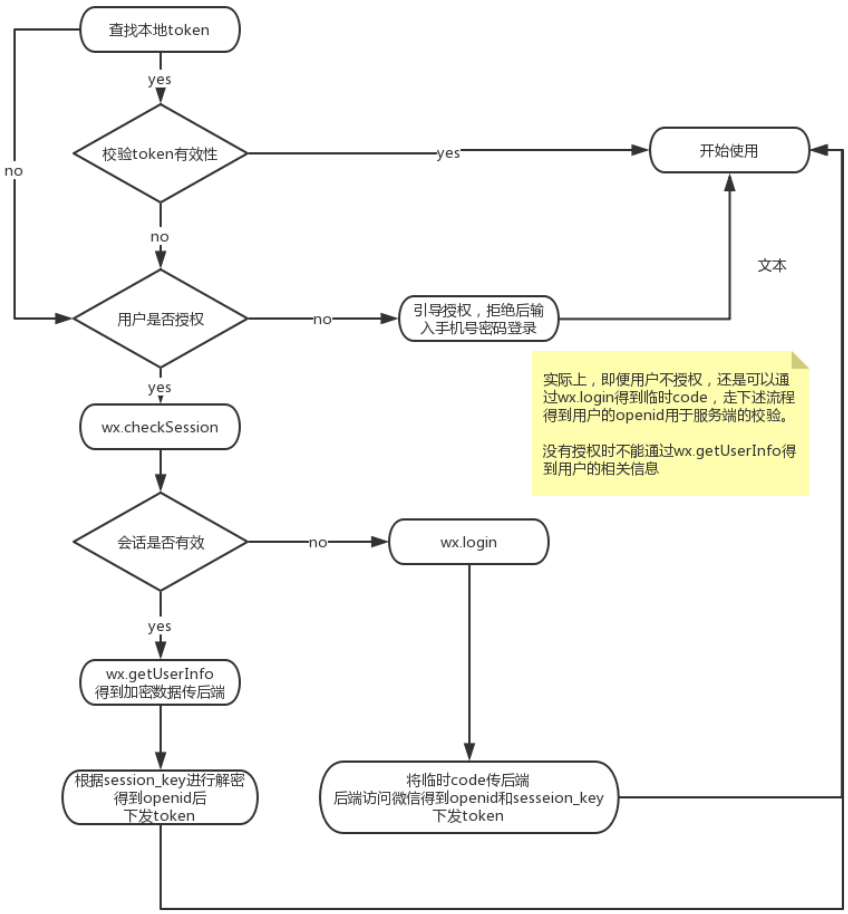
- **沿用之前的VSCODE开发工具，只在微信开发工具中使用调试功能。开发环境引入eslint语法检测。后面视情况决定是否使用第三方开发工具如webpack等。**

后续

- 建议启动小程序的开发流程
- TODO

  如果第三方通过小程序得到了我们的源码，掌握了我们的交易流程，是否会有安全漏洞?

进一步梳理交易流程中的异常分支
小程序登录势必需要对微信账号和笑联账号进行绑定



a. 小程序端查找保存的token。如果有则查看token是否有效。如果有效，直接使用token调用服务。如果无效，开始获取token流程。
b. 如果用户没有授权，引导用户走授权流程。如果用户拒绝，让用户输入手机号和密码登录。
c. 如果用户已授权，先到微信服务器checksession。如果有效则getuserinfo。得到后加密数据将之传给后端解密，后端查找对应的openid来下发token，进入正常使用流程。
d. 如果checksession失效，走如下流程：用户端wx.login->获取临时code传给服务端—>服务端到微信服务器校验此code ->服务端获得用户openid和session_key->服务端更新session_key，同时下发token。

```
// 服务器请求微信验证code的请求，url如下，appid和secret分别为小程序的固有参数。
wx.request({
      url: `https://api.weixin.qq.com/sns/jscode2session?appid=${appId}&secret=${secret}&js_code=${code}&grant_type=authorizati
      data: {},
      method: "POST",
      header: {
        "content-type": "application/json" // 默认值
      },
      success: function(res) {
        console.log(res)
        // res格式：  {"session_key":"zL\/2HyaxuZFpOnvJrOXHKQ==","openid":"oEPFJ5B1UH8VkmpRW_dNn6L1QlJ4"}
      },
      complete: function(res) {
        console.log("complete", res)
      }
    })
```

小程序和原生在不同交易阶段的交叉登录处理(比如现在已经发现的一个案例: 在一端开阀使用后，在另一端是无法获得关阀命令的，在设备关阀前也无法连接设备。只能等关阀后才能连上设备。)
允许用户同时登录小程序和原生应用么？如果用户登录了小程序，推送是否能送达？

无标签