# 手把手教会你小程序登录鉴权

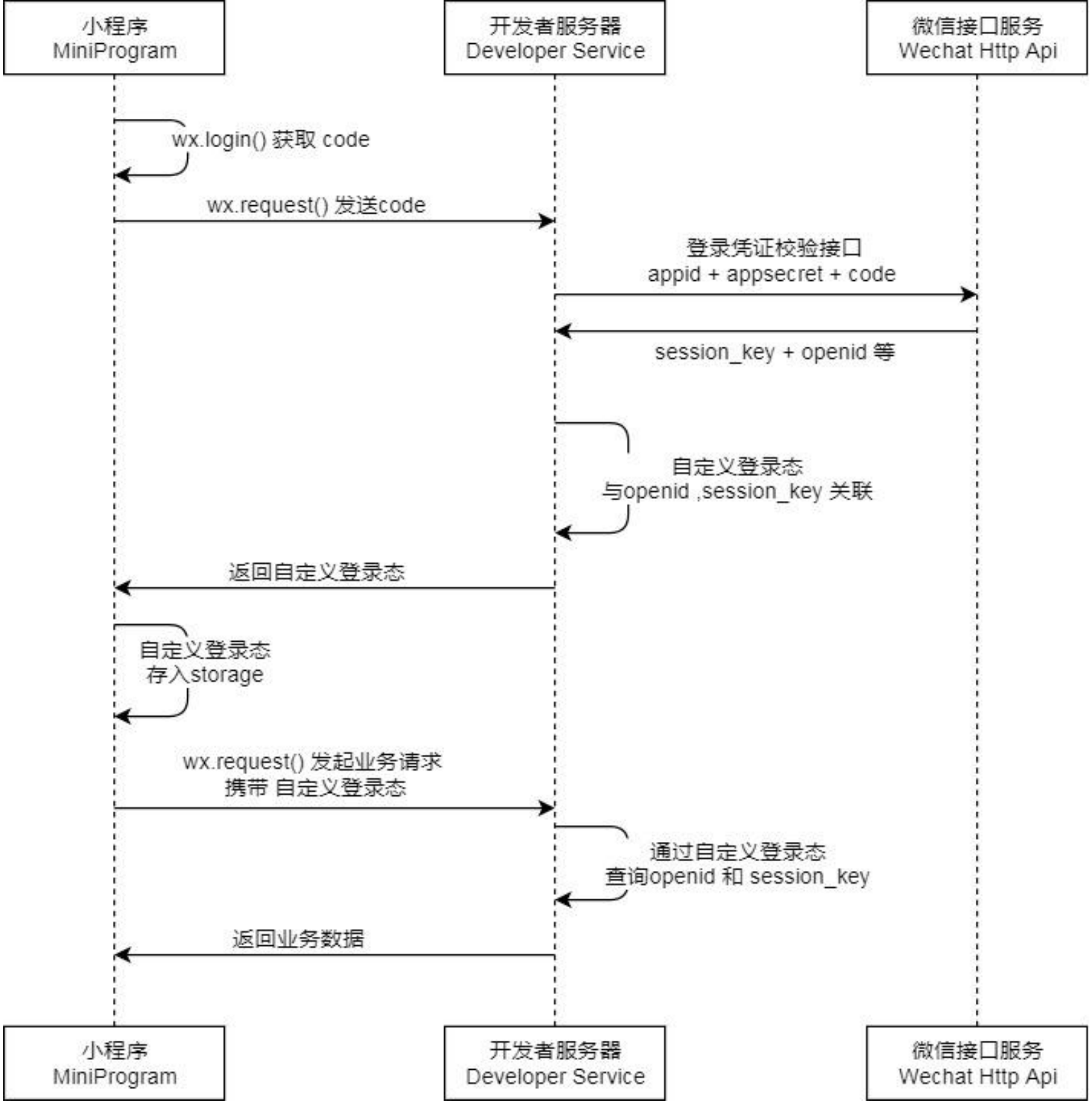
2018 年 04 月 08 日

## 导语

为了方便小程序应用使用微信登录态进行授权登录，微信小程序提供了登录授权的开放接口。乍一看文档，感觉文档上讲的非常有道理，但是实现起来又真的是摸不着头脑，不知道如何管理和维护登录态。本文就来手把手的教会大家在业务里如何接入和维护微信登录态。

## 接入流程

这里官方文档上的流程图已经足够清晰，我们直接就该图展开详述和补充。




首先大家看到这张图，肯定会注意到小程序进行通信交互的不止是小程序前端和我们自己的服务端，微信第三方服务端也参与其中，那么微信服务端在其中扮演着怎样的角色呢？我们一起来串一遍登录鉴权的流程就明白了。

### 1. 调用wx.login生成code

wx.login()这个API的作用就是为当前用户生成一个临时的登录凭证，这个临时登录凭证的有效期只有五分钟。我们拿到这个登录凭证后就可以进行下一步操作：获取openid和session_key

```
wx.login({
    success: function(loginRes) {
```

```
        if (loginRes.code) {
            // example: 081LXytJ1Xq1Y40sg3uJ1FWntJ1LXyth
        }
    }
});
```

### 2. 获取openid和session_key

我们先来介绍下openid，用过公众号的童鞋应该对这个标识都不陌生了，在公众平台里，用来标识每个用户在订阅号、服务号、小程序这三种不同应用的唯一标识，也就是说每个用户在每个应用的openid都是不一致的，所以在小程序里，我们可以用openid来标识用户的唯一性。

那么session_key是用来干嘛的呢？有了用户标识，我们就需要让该用户进行登录，那么session_key就保证了当前用户进行会话操作的有效性，这个session_key是微信服务端给我们派发的。也就是说，我们可以用这个标识来间接地维护我们小程序用户的登录态，那么这个session_key是怎么拿到的呢？我们需要在自己的服务端请求微信提供的第三方接口`https://api.weixin.qq.com/sns/jscode2session`，这个接口需要带上四个参数字段：

| 参数 | 值 |
| --- | --- |
| appid | 小程序的appid |
| secret | 小程序的secret |
| js_code | 前面调用wx.login派发的code |
| grant_type | 'authorization_code' |

从这几个参数，我们可以看出，要请求这个接口必须先调用wx.login()来获取到用户当前会话的code。那么为什么我们要在服务端来请求这个接口呢？其实是出于安全性的考量，如果我们在前端通过request调用此接口，就不可避免的需要将我们小程序的appid和小程序的secret暴露在外部，同时也将微信服务端下发的session_key暴露给“有心之人”，这就给我们的业务安全带来极大的风险。除了需要在服务端进行session_key的获取，我们还需要注意两点：

- session_key和微信派发的code是一一对应的，同一code只能换取一次session_key。每次调用wx.login()，都会下发一个新的code和对应的session_key，为了保证用户体验和登录态的有效性，开发者需要清楚用户需要重新登录时才去调用wx.login()

- session_key是有失效性的，即便是不调用wx.login，session_key也会过期，过期时间跟用户使用小程序的频率成正相关，但具体的时间长短开发者和用户都是获取不到的

```
function getSessionKey (code, appid, appSecret) {
    var opt = {
        method: 'GET',
        url: 'https://api.weixin.qq.com/sns/jscode2session',
        params: {
            appid: appid,
            secret: appSecret,
            js_code: code,
            grant_type: 'authorization_code'
        }
    };
    return http(opt).then(function (response) {
        var data = response.data;
        if (!data.openid || !data.session_key || data.errcode) {
            return {
                result: -2,
                errmsg: data.errmsg || '返回数据字段不完整'
            }
        } else {
            return data
        }
    });
}
```

### 3. 生成3rd_session

前面说过通过session_key来“间接”地维护登录态，所谓间接，也就是我们需要**自己维护用户的登录态信息**，这里也是考虑到安全性因素，如果直接使用微信服务端派发的session_key来作为业务方的登录态使用，会被“有心之人”用来获取用户的敏感信息，比如wx.getUserInfo()这个接口呢，就需要session_key来配合解密微信用户的敏感信息。

那么我们如果生成自己的登录态标识呢，这里可以使用几种常见的不可逆的哈希算法，比如md5、sha1等，将生成后的登录态标识（这里我们统称为'skey'）返回给前端，并在前端维护这份登录态标识(一般是存入storage)。而在服务端呢，我们会把生成的skey存在用户对应的数据表中，前端通过传递skey来存取用户的信息。

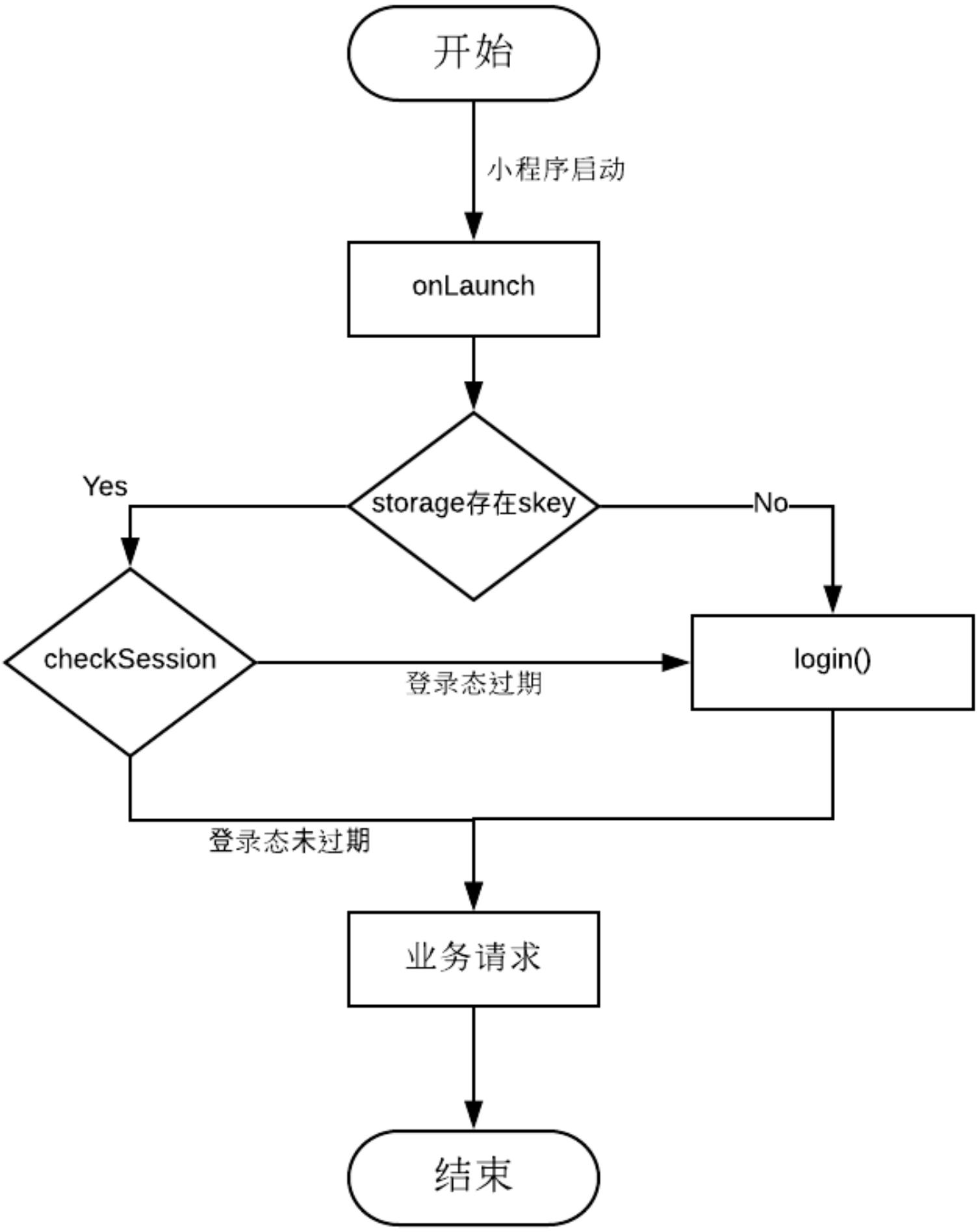
可以看到这里我们使用了sha1算法来生成了一个skey：

```
const crypto = require('crypto');

return getSessionKey(code, appid, secret)
    .then(resData => {
        // 选择加密算法生成自己的登录态标识
        const { session_key } = resData;
        const skey = encryptSha1(session_key);
    });

function encryptSha1(data) {
    return crypto.createHash('sha1').update(data, 'utf8').digest('hex')
}
```

**4. checkSession**

前面我们将skey存入前端的storage里，每次进行用户数据请求时会带上skey，那么如果此时session_key过期呢？所以我们需要调用到wx.checkSession()这个API来校验当前session_key是否已经过期，这个API并不需要传入任何有关session_key的信息参数，而是微信小程序自己去调自己的服务来查询用户最近一次生成的session_key是否过期。如果当前session_key过期，就让用户来重新登录，更新session_key，并将最新的skey存入用户数据表中。

checkSession这个步骤呢，我们一般是放在小程序启动时就校验登录态的逻辑处，这里贴个校验登录态的流程图：




下面代码即校验登录态的简单流程:

```
let loginFlag = wx.getStorageSync('skey');
if (loginFlag) {
    // 检查 session_key 是否过期
    wx.checkSession({
        // session_key 有效(未过期)
        success: function() {
```

```
            // 业务逻辑处理
        },

        // session_key 过期
        fail: function() {
            // session_key过期，重新登录
            doLogin();
        }
    });
) else {
    // 无skey，作为首次登录
    doLogin();
}
```

## 5. 支持emoji表情存储

如果需要将用户微信名存入数据表中，那么就确认数据表及数据列的编码格式。因为用户微信名可能会包含emoji图标，而常用的UTF8编码只支持1-3个字节，emoji图标刚好是4个字节的编码进行存储。

这里有两种方式(以mysql为例)：

### 1.设置存储字符集

在mysql5.5.3版本后，支持将数据库及数据表和数据列的字符集设置为utf8mb4，因此可在/etc/my.cnf设置默认字符集编码及服务端编码格式

```
// my.cnf

[client]
default-character-set=utf8mb4
[mysql]
default-character-set=utf8mb4
[mysqld]
character-set-client-handshake = FALSE
character-set-server=utf8mb4
collation-server=utf8mb4_unicode_ci
```

设置完默认字符集编码及服务端字符集编码，如果是对已经存在的表和字段进行编码转换，需要执行下面几个步骤：

- 设置数据库字符集为utf8mb4

```
ALTER DATABASE 数据库名称 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_unicode_c
```

- 设置数据表字符集为utf8mb4

```
ALTER TABLE 数据表名称 CONVERT TO CHARACTER SET utf8mb4 COLLATE utf8mb4_unicc
```

- 设置数据列字段字符集为utf8mb4

```
ALTER TABLE 数据表名称 CHANGE 字段列名称 VARCHAR(n) CHARACTER SET utf8mb4 COLL
```

这里的COLLATE指的是排序字符集，也就是用来对存储的字符进行排序和比较的，utf8mb4常用的collation有两种：utf8mb4_unicode_ci和utf8mb4_general_ci，一般建议使用utf8mb4_unicode_ci，因为它是基于标准的Unicode Collation Algorithm(UCA)来排序的，可以在各种语言进行精确排序。这两种排序方式的具体区别可以参考：What's the difference between utf8_general_ci and utf8_unicode_ci

2. 通过使用sequelize对emoji字符进行编码入库，使用时再进行解码

这里是sequelize的配置，可参考Sequelize文档

```
{
        dialect: 'mysql',     // 数据库类型
        dialectOptions: {
          charset: 'utf8mb4',
          collate: "utf8mb4_unicode_ci"
       },
}
```

## 最后

前面讲了微信小程序如何接入微信登录态标识的详细流程，那么如何获取小程序中的用户数据以及对用户敏感数据进行解密，并保证用户数据的完整

性，我将在下一篇文章给大家做一个详细地介绍。

## 团队开源

> 腾讯IVWEB团队的工程化解决方案feflow已经开源：Github主页：feflow

> 如果对您的团队或者项目有帮助，请给个Star支持一下哈~