17 坏人请走开：开发实现用户成长体系风控规则

更新时间：2019-07-29 15:35:40

> 横眉冷对千夫指，俯首甘为孺子牛。
>
> ——鲁迅

在前面几节我们已经完成了用户成长体系的大部分功能开发，本节我们一起来完成用户成长体系的最后一块拼图，风控规则。

在本章第一节和第二节我们已经详细梳理了用户成长体系的风控规则*：

- 风控规则 1：单日最大同步微信运动步数获得的成长值上限为 100000（正常用户单日走路超过十万步的几率很小，单日微信运动步数大于十万可以认为是异常行为）
- 风控规则 2：每个用户的每次成长值变动都需要进行前面两条规则的判断，如果触发规则，自动锁定用户账号，同时在风控日志中记录这次用户账号锁定操作及锁定原因
- 风控规则 3：被锁定用户在打开小程序任何页面时，自动跳转到账号锁定提示页面，以此来实现禁止用户执行任何操作

*发表笔记获得成长值的风控逻辑与微信运动步数获得成长值的风控逻辑完全一致，本节讲解微信运动步数获得成长值的风控逻辑，发表笔记获得成长值的风控逻辑请阅读本专栏源代码。

在实现风控规则之前，我们需要先实现用户首次打开小程序自动注册的功能，该功能即在 `user` 表中添加该用户的记录。实现该功能后，在触发风控规则时我们才能设置该用户的锁定标志 `isLocked` 。

1. 用户首次打开小程序自动注册

在第三章第三节我们已经介绍了使用云开发可以非常简单地实现小程序用户的自动登录。

要实现用户首次打开小程序自动注册，我们需要在 app.js 中调用自动登录云函数，并在自动登录云函数中增加在 `user` 表中添加该用户的功能。

此外，该云函数还应该返回用户的账号锁定标志 `isLocked` ，这样才能实现风控规则 3。

云函数 `login` 的完整实现如下：

```
// 云函数入口文件
// 部署：在 cloud-functions/login 文件夹右击选择 “上传并部署”
const cloud = require('wx-server-sdk')

// 初始化 cloud
cloud.init()

// 云函数入口函数
/**
 * 获取当前用户的OpenId、锁定标志并自动注册
 * @return {object} 用户信息
 * {
 *    openid, //当前用户的openid
 *    isLocked //当前用户的锁定标志
 * }
 */
exports.main = async(event, context) => {
  // 获取 WX Context (微信调用上下文)，包括 OPENID、APPID、及 UNIONID（需满足 UNIONID 获取条件）
  const wxContext = cloud.getWXContext()

  const db = cloud.database()
  const _ = db.command

  //查询用户是否已注册
  var queryResult = (await db.collection('user')
    .where({
      //云函数是在服务端操作，对所有用户的数据都有操作权限
      //在云函数中查询用户数据，需要添加openid的查询条件
      _openid: wxContext.OPENID
    })
    .count()).total
  //如果用户还未注册，自动注册（即在 user 表中添加该用户信息）
  if (queryResult <= 0) {
    await db.collection('user').add({
      data: {
        _openid: wxContext.OPENID, //云函数添加数据不会自动插入openid，需要手动定义
        date: db.serverDate(),
        growthValue: 0, //成长值
        isLocked: false //是否因触发风控规则被锁定
      }
    })
  }

  //获取当前用户信息
  var userInfo = (await db.collection('user')
  .where({
    _openid: wxContext.OPENID
  })
  .get()).data[0]

  //返回用户的openid与锁定标志
  return {
    openid: wxContext.OPENID,
    isLocked: userInfo.isLocked
  }
}
```

app.js 在用户打开小程序时执行，任何用户打开小程序都将先执行 `onLaunch` 事件的代码：

```
/* 参考https://developers.weixin.qq.com/miniprogram/dev/reference/api/App.html */
App({
  /**
   * 生命周期回调—监听小程序初始化。
   */
  onLaunch: function () {
    //调用云函数获取用户openid，用户锁定标志，如用户未注册将自动注册
    wx.cloud.callFunction({
      name: 'login',
      data: {},
      success: res => {
        this.globalData.isLocked = res.result.isLocked
        this.globalData.openid = res.result.openid
      }
    })
  },
  //小程序全局变量
  globalData: {
    openid: null,
    isLocked: null
  }
})
```

2. 实现风控规则校验

风控规则是在服务器端运行，应该在云函数中实现。

风控规则云函数在用户获取成长值时调用，在本章第三节已经给出了在何处调用风控规则的示例。

根据风控规则1、风控规则2、风控规则3，我们可以整理出风控规则云函数的程序逻辑：

- 步骤 1 ：定义单日最大同步微信运动步数获得的成长值上限为 100000；
- 步骤 2 ：从成长值获取记录表 `user_growth_value` 中获取当前用户的所有记录 `allGrowthValueRecords` ，由于云函数端每次查询数据库返回的记录数量最多为 100 条，而用户的成长值获取记录数可能超过 100 条，因此当前用户的所有记录需要分多次读取；
- 步骤 3 ： 建立一个 key-value 类型的数组 `addGrowthValueGroupedByDate` ，用于统计每天用户获得的成长值总数。循环读取 `allGrowthValueRecords` 中的每一条记录，统计每天用户获得的成长值总数，并记录到数组 `addGrowthValueGroupedByDate` 中，key是日期，value是用户当日获得的成长值总数；
- 步骤 4：依次判断 `addGrowthValueGroupedByDate` 中的用户每天获得的成长值总数是否超过成长值上限，如果某天用户获得的成长值总数超过成长值上限，则在风控异常日志表 `risk` 中记录该异常，并将用户在 `user` 表中的锁定标志 `isLock` 设置为 true 。

风控规则在云函数 `growthValueRiskControl` 的 index.js 中具体实现：

```
// 初始化 cloud
const cloud = require('wx-server-sdk')
cloud.init()
const db = cloud.database()
const _ = db.command

//在云函数中每次最多返回 100 条数据
const MAX_LIMIT = 100

//步骤 1 ：定义单日最大同步微信运动步数获得的成长值上限为 100000
const MAXSYNCWERUNPERDAY = 100000

// 云函数入口函数
/**
 * 成长值风控规则
 * @param {data} 要进行风控规则检查的用户
 * {
 *    data:{
 *      //由于本云函数是由服务端调用，而非小程序客户端调用，
```

```
 *      //cloud.getWXContext()无法获取openid，openid 需要作为参数传入
 *      openid
 *    }
 *  }
 * @return {object} 风控规则运行结果
 * {
 *    data,    //bool true表示触发风控规则，用户账户锁定并记录风控日志 false表示正常
 * }
 */
exports.main = async(event, context) => {

//----------------------步骤 2 分多次读取，获取当前用户的所有成长值获取记录----------------begin
  // 先取出集合记录总数
  const countResult = await db.collection('user_growth_value').where({
    _openid: event.openid
  }).count()
  const total = countResult.total
  // 计算需分几次取
  const batchTimes = Math.ceil(total / MAX_LIMIT)
  // 承载所有读操作的 promise 的数组
  const tasks = []
  for (let i = 0; i < batchTimes; i++) {
    const promise = db.collection('user_growth_value').where({
      _openid: event.openid
    }).skip(i * MAX_LIMIT).limit(MAX_LIMIT).get()
    tasks.push(promise)
  }
  // 等待所有
  var allGrowthValueRecords = (await Promise.all(tasks)).reduce((acc, cur) => ({
    data: acc.data.concat(cur.data),
    errMsg: acc.errMsg,
  }))
//----------------------步骤 2 分多次读取，获取当前用户的所有成长值获取记录----------------end

//----------------------步骤 3 统计每天用户从微信运动步数获得的成长值总数------------------begin
  //key-value 类型的数组，用于统计每天用户获得的成长值总数
  //key是日期，value是用户当日获得的成长值总数
  var addGrowthValueGroupedByDate = new Array()
  //循环读取的每一条成长值获取记录，统计每天用户获得的成长值总数
  for (var i in allGrowthValueRecords.data) {
    var item = allGrowthValueRecords.data[i]
    //在微信运动步数获取成长值的风控规则中，只统计微信运动的记录
    if (item.operation == "微信运动") {
      var timestamp = item.timestamp
      if (addGrowthValueGroupedByDate[timestamp] !== undefined) {
        //如果已存在当日的成长值，则累加当日成长值
        addGrowthValueGroupedByDate[timestamp] += item.changeGrowthValue
      } else {
        //如果不存在当日成长值，则新增当日成长值统计记录
        addGrowthValueGroupedByDate[timestamp] = item.changeGrowthValue
      }
    }
  }
//----------------------步骤 3 统计每天用户从微信运动步数获得的成长值总数------------------end

//----------------------步骤 4 校验用户每天获得的成长值总数是否超过成长值上限----------------begin
  //返回值，是否触发风控规则
  var result = false
  //循环校验用户每天获得的成长值总数是否触发风控规则
  for (var key in addGrowthValueGroupedByDate) {
    if (addGrowthValueGroupedByDate[key] > MAXSYNCWERUNPERDAY) {
      //如果用户每天获得的成长值总数是否超过成长值上限
      //触发风控规则，锁定用户账号
      await db.collection('user')
        .where({
          _openid: event.openid
        })
        .update({
          data: {
            isLocked: true //修改用户锁定标志为锁定状态
          }
        })
      //记录风控异常日志
```

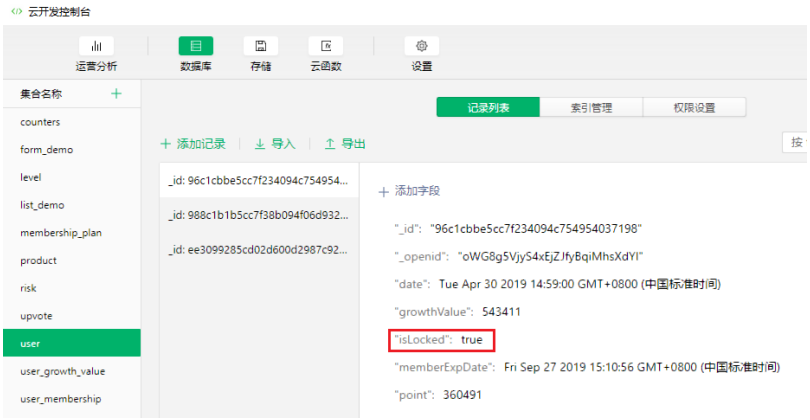
```
      await db.collection('risk')
        .add({
          data: {
            _openid: event.openid, //用户OpenID
            date: db.serverDate(), //检查到异常的时间
            event: '微信运动', //检查到的异常类型
            maxLimit: MAXSYNCWERUNPERDAY, //触发的风控规则上限值
            riskTime: key, //哪一天获得的成长值超过上限
            riskVaule: addGrowthValueGroupedByDate[key] //当天获得了多少超限成长值
          }
        })
      //设置返回值为触发风控规则
      result = true
    }
  }
//----------------------步骤 4 校验用户每天获得的成长值总数是否超过成长值上限----------------end

  //返回风控规则校验结果
  return {
    data: result
  }
}
```

在实现风控规则后，结合本章第三节实现的同步微信运动步数获得成长值代码，我们可以测试风控规则是否生效。

首先将 `MAXSYNCWERUNPERDAY` 修改为肯定会触发风控规则的值，例如100，然后在微信开发者工具的模拟器中点击本章第三节页面中的同步按钮。在同步完成后，即可在数据库中查看到用户表 `user` 中自己的锁定标志 `isLock` 已经变为锁定状态 `true`，如图 6 所示。同时，在风控异常日志表 `risk` 中可以查看到异常日志记录。

图 6 用户表锁定标志数据库记录



在测试完成后，别忘了把成长值上限值恢复为 100000，并在云数据库中修改自己的锁定标志为 `false` 。

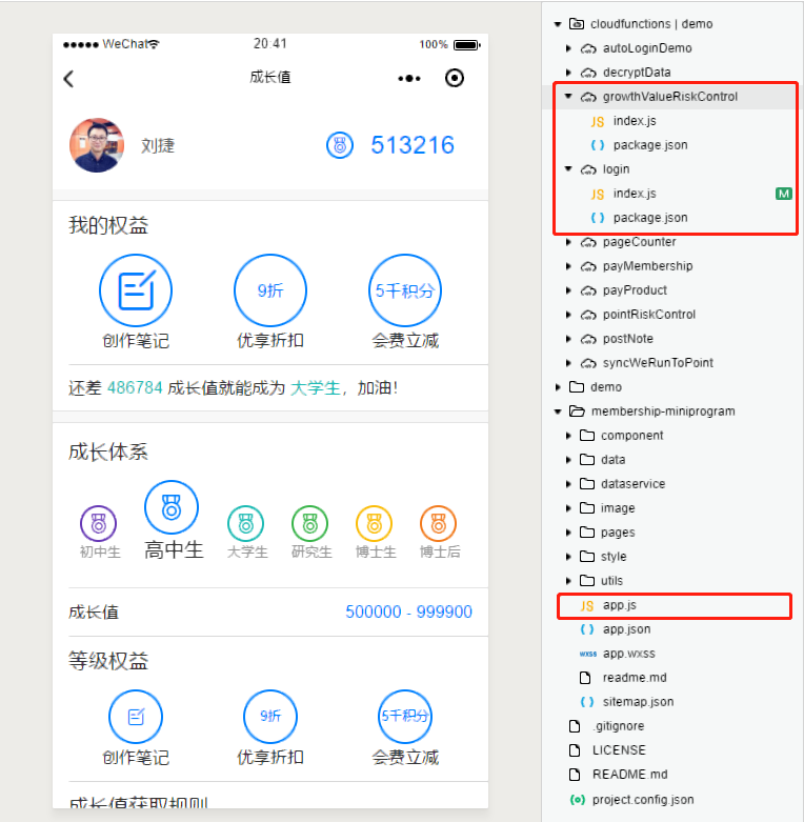
3. 专栏源代码

本专栏源代码已上传到 GitHub ，请访问以下地址获取：

https://github.com/liujiec/Membership-ECommerce-Miniprogram

本节源代码内容在图 7 红框标出的位置。

图 7 本节源代码位置



下节预告

从下一节开始，我们将实现积分体系，首先完成积分体系的业务设计。

实践环节

实践是通往大神之路的唯一捷径。

本节实操内容：

- 参照本节内容，在自己的云开发环境中编写风控规则云函数，并测试风控规则是否生效。

16 开发实现用户成长体系页面

业务设计：积分就是钱，千万别太随便了

精选留言 0

欢迎在这里发表留言，作者筛选后可公开显示

目前暂无任何讨论