09 云函数与表单校验组件让表单提交如此简单

更新时间：2019-07-11 15:01:21

> 人的差异在于业余时间。
>
> ——爱因斯坦

上一节我们学习了使用云开发数据库实现一个带数据交互的页面。

前面两小节的内容主要集中在向用户显示内容，在小程序开发中还有一个常用场景-输入界面，即用户输入内容。

本节，我们就来讲解如何编写供用户输入数据的表单页面、如何验证表单输入数据是否正确、如何提交表单以及使用云开发中的云函数快速实现自动登录能力。

第一个实践内容是表单，如图 15 所示。

图 15 表单Demo

1. 表单页面

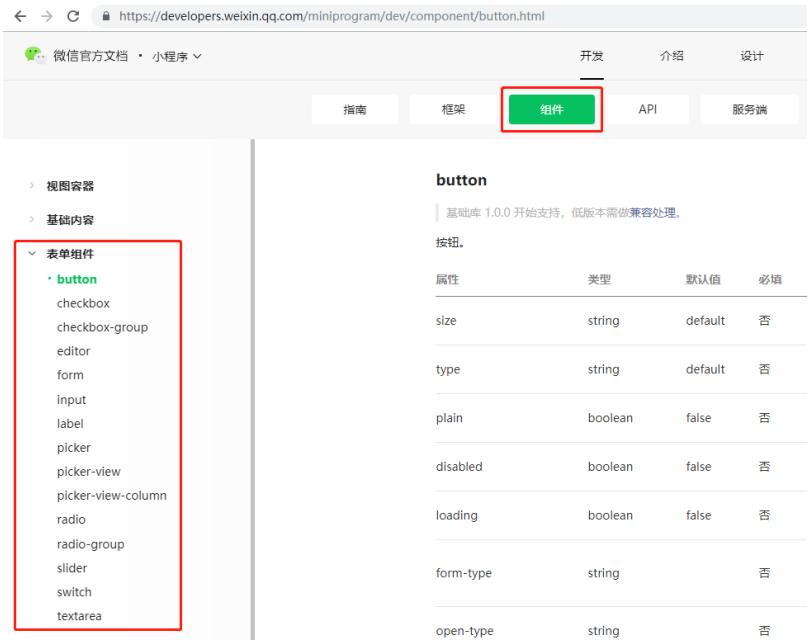
用户输入内容主要分为两类：

- 一类是填写个人信息，如注册表单等。这类表单包含文本框- input、单选列表项- radio 、复选列表项- checkbox 等（见图 15）；
- 一类是发表评论、动态，填写个人简介等内容。这类表单主要包含文本域- textarea （见图 15）。

在小程序官方组件中已经包含了开发中常用的各种表单输入组件。**小程序表单组件在微信官方的“小程序开发文档”中有详细介绍，文档位置如下：**

- 入口网址：小程序表单组件，如果微信官方文档改版导致链接失效，请按图索骥。
- 入口位置：在“小程序开发文档”首页，如图 16 红框标出部分。

图 16 小程序表单组件文档位置



同时，在“小程序开发文档”的“体验小程序”一节中也有这些组件的小程序Demo及源代码（见本章第一节图 8）。

而WeUI中则提供了表单页面布局的具体方案，包括 button 、list 、input 等组件，如图 17 所示。WeUI的小程序Demo及源代码请回顾本章第一节 2.1 UI框架。

图 17 WeUI表单布局组件

因此，学习小程序表单开发的第一步，是先阅读“小程序开发文档”中表单组件的内容，结合小程序组件 Demo 、WeUI Demo及其源代码，学会各个表单组件的使用语法和表单界面布局。

另外，本专栏也编写了一个完整的表单Demo供各位同学参考，源代码位置见本节“ 6. 专栏源代码”。

对一些需要特殊处理的表单组件，在这里选取本专栏源代码中的对应代码片段做一个简要说明，方便各位同学理解。

1.1 密码输入框

密码输入框需要隐藏用户数据的密码，这需要在 input 组件中设置 `password="true"` 。

```
<input name="password" value="{{ password }}" type="text" password="true" />
```

1.2 单选列表项- radio

radio 组件需要首先在 JS 逻辑的 data 中设置每个选项的值 `radio` 。同时定义一个数据 `gender` 用于保存用户的选择。这样当我们进行表单校验、或向云数据库提交表单数据时就可以直接使用 `gender` ，而不必每次都去查找 radio 组件中哪个选项的 `checked=true` ：

```
  /**
   * 页面的初始数据
   */
  data: {
    gender: '',
    radio: [{
        name: '男',
        value: 'male',
        checked: !1,
      },
      {
        name: '女',
        value: 'female',
      },
    ],
  },
```

然后在 WXML 页面模板中使用 “列表渲染”（语法见第二节 4.4 实现页面显示列表数据），并复制WeUI 中 input 组件提供的样式显示每个选项，同时需要绑定一个事件 `bindchange="radioChange"` 来响应用户点击选项的动作：

```
<radio-group name="gender" bindchange="radioChange">
    <label class="weui-cell weui-check__label" wx:for="{{radio}}" wx:key="value">
        <radio class="weui-check" value="{{item.value}}" checked="{{item.checked}}" />
        <view class="weui-cell__bd">{{item.name}}</view>
        <view class="weui-cell__ft weui-cell__ft_in-radio" wx:if="{{item.checked}}">
            <icon class="weui-icon-radio" type="success_no_circle" size="16"></icon>
        </view>
    </label>
</radio-group>
```

最后还需要在 JS 逻辑编写用户点击选项的事件函数 `radioChange` ，来实现用户点击的选项被选中的效果：

```
  /**
   * radio改变时刷新data中的值
   */
  radioChange(e) {
    const value = e.detail.value
    const radio = this.data.radio
    //Object.assign是ES6语法，感兴趣的同学可在网上搜索ES6具体了解
    //其他同学只需要记住这样写实现的操作是：
    //将用户当前选中的选项设置checked=true，其他选项设置checked=false
    //当然这个操作也可以用for循环实现，这作为一个实践作业，请各位同学尝试改写为for循环实现
    const items = radio.map(n => {
      return Object.assign({}, n, {
        checked: n.value === value,
      })
    })
    this.setData({
      radio: items,
      gender: value,
    })
  },
```

小程序每个组件都有自己的事件定义，在“小程序开发文档”中类型是 `EventHandle` 的属性就是事件。

要完整地理解小程序的事件系统，请参阅微信官方“小程序开发文档”的“指南”->“小程序框架”->“视图层”->“事件系统”小节。

1.3 复选列表项- checkbox

checkbox 组件的使用方法类似 radio ，实现代码请阅读本专栏源代码 `demo/pages/06_form` 中的“技能（复选列表项-checkbox）”（页面位置见图 15）。

1.4 选择- picker

picker 组件有多种用法，详细介绍请阅读“小程序开发文档”中的介绍、小程序组件Demo源代码及本专栏源代码。

这里仅介绍最常用的时间选择器的使用方法。

首先在 JS 逻辑的 data 中定义一个数据 `date` 用于保存用户选择的时间：

```
/**
 * 页面的初始数据
 */
data: {
  date: '',
},
```

然后在 WXML 页面模板中复制 WeUI 中 input 组件提供的样式，并设置 picker 组件为时间选择器 `mode="date"` ，同时需要绑定一个事件 `bindchange="bindDateChange"` 来响应用户选择时间的动作：

```
<view class="weui-cell__bd">
    <picker name="date" mode="date" value="{{date}}" start="2015-09-01" end="2017-09-01" bindchange="bindDateChange">
        <view class="weui-input">{{date}}</view>
    </picker>
</view>
```

最后还需要在 JS 逻辑 编写用户选择时间的事件函数 `bindDateChange` ，来记录并在界面中显示用户选择的时间：

```
/**
 * 出生日期改变时刷新data中的值
 */
bindDateChange: function(e) {
  this.setData({
    date: e.detail.value
  })
},
```

2. 表单验证

在我们完成表单页面的显示后，还需要对用户输入的表单数据进行校验及错误提示，比如对必须输入内容的组件判断用户是否未输入内容，如用户未输入内容在界面中提示用户该组件是必填项。

微信官方并未提供表单验证功能，但目前在 **GitHub** 上已经有了一些开源的小程序表单验证组件。其中一个比较完善的表单验证组件是**"WxValidate - 表单验证"**。

该组件的源代码可从如下地址获取： https://github.com/skyvow/wx-extend，在该组件的 GitHub 项目主页有组件的使用介绍。

同时，本专栏的源代码中也包含了该组件，位于 `demo/pages/utils/validate` 目录中。该组件的完整使用示例包含在本专栏源代码的 `demo/pages/06_form` 中，本专栏的示例对源项目的错误提示做了改进，结合了 **WeUI** 组件 **input** 中的错误提示样式。

图 18 本专栏示例的表单错误提示效果

这里以实现“姓名”这个文本框的表单校验为例来介绍表单校验的实现方式，如何完整实现如图 18 所示的表单校验效果，请阅读本专栏源代码。

- 2.1 首先需要在 JS 逻辑中引用表单验证组件，见 JS 逻辑源代码注释 2.1 部分；
- 2.2 设置页面的初始数据，见 JS 逻辑源代码注释 2.2 部分；
- 2.3 在页面加载时（即 `onLoad` 事件中）设置表单的验证规则 `rules` 和 验证规则对应的错误提示信息 `messages`，见 JS 逻辑源代码注释 2.3 部分；
- 2.4 定义表单提交事件 `submitForm` 在用户点击提交表单按钮式进行表单校验。表单校验的具体流程为：首先调用表单验证组件，如果表单验证组件返回验证失败，则在 `errorMap` 中记录表单验证出的所有错误信息，在 `errorMap` 中包含错误信息的输入组件会将自己的标题设置为红色；第二步是将错误信息的第一条内容记录到 `error` 中作为在页面顶部显示的错误信息内容，然后在页面顶部显示3秒错误信息内容；如果表单验证组件返回验证成功，弹出提示框提示表单验证成功，见 JS 逻辑源代码注释 2.4 部分。

JS 逻辑源代码

```
//2.1
//引用表单验证组件"WxValidate - 表单验证"
import WxValidate from '../../utils/validate/WxValidate.js'
//2.1

Page({

//2.2
  /**
   * 设置页面的初始数据
   */
  data: {
    name: '', //记录用户输入的姓名
    error: "", //页面顶部的错误提示显示内容
    errorMap: {}, //记录表单验证发现的所有错误，用于界面显示错误提示
  },
//2.2
```

```
//2.3
  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function(options) {
    //在页面加载时设置表单的验证规则
    this.initValidate()
  },

  /**
   * 设置表单的验证规则
   */
  initValidate() {
    //验证规则
    const rules = {
      name: {
        required: true, //必填字段
        minlength: 2,   //最小长度为2个字
        maxlength: 4    //最大长度为4个字
      },
    }

    // 验证规则对应的错误提示信息
    const messages = {
      name: {
        required: '请输入姓名',
        minlength: '姓名最少2个字',
        maxlength: '姓名最多4个字'
      },
    }

    // 创建"WxValidate - 表单验证"组件的实例对象
    this.WxValidate = new WxValidate(rules, messages)
  },
//2.3

//2.4
  /**
   * 点击提交表单时进行表单验证
   */
  submitForm(e) {
    const params = e.detail.value

    // 传入表单数据，调用表单验证方法
    if (!this.WxValidate.checkForm(params)) {
      //如果表单验证失败，记录表单验证出的所有错误信息
      this.setErrorMap(this.WxValidate.errorList)
      //设置页面顶部要显示的错误信息内容
      this.setData({
        error: this.WxValidate.errorList[0].msg
      })
      //在页面顶部显示3秒错误信息内容
      this.showTopTips()
      return false
    }

    //如果表单验证成功，进行提示
    wx.showModal({
      title: '验证结果',
      content: '验证成功'
    })
  },

  /**
   * 记录表单验证出的所有错误信息，供页面将输入错误区域的标题设置为红色
   */
  setErrorMap: function(e) {
    var map = {}
    for (var index in e) {
      map[e[index].param] = e[index].msg.length > 0
    }
    this.setData({
      errorMap: map
```

```
    })
  },

  /**
   * 在页面顶部显示3秒错误信息内容
   */
  showTopTips: function() {
    var that = this;
    this.setData({
      showTopTips: true
    });
    setTimeout(function() {
      that.setData({
        showTopTips: false
      });
    }, 3000);
  },
//2.4
})
```

- 2.5 在 WXML 页面模板添加页面顶部的错误提示区域，见 WXML 页面模板源代码 2.5 部分；
- 2.6 在 WXML 页面模板添加一个表单组件 form，并设置它的表单提交事件为 2.4 中定义的事件。另外添加一个表单提交按钮，以触发表单提交事件。见 WXML 页面模板源代码 2.6 部分；
- 2.7 在 WXML 页面模板添加一个姓名输入框，并添加根据表单校验结果将输入错误区域的标题设置为红色的判断 `{{ errorMap['name']?'weui-cell_warn':'' }}`，见 WXML 页面模板源代码 2.7 部分。

WXML 页面模板源代码

```
<!-- 2.5 页面顶部的错误提示区域 -->
<view class="weui-toptips weui-toptips_warn" wx:if="{{showTopTips}}">{{error}}</view>

<!-- 2.6 表单组件 -->
<form bindsubmit="submitForm">

  <!-- 2.7 姓名输入框 -->
  <view class="weui-cells">
    <view class="weui-cell weui-cell_input">
      <!-- 2.7 根据表单校验结果将输入错误区域的标题设置为红色 -->
      <view class="weui-cell__hd  {{ errorMap['name']?'weui-cell_warn':'' }}">
        <view class="weui-label">姓名</view>
      </view>
      <view class="weui-cell__bd">
        <input name="name" value="{{ name }}" class="weui-input"
        type="text" placeholder="请输入2-4个字的姓名" />
      </view>
    </view>
  </view>

  <!-- 2.6 表单提交按钮 -->
  <view class="weui-btn-area">
    <button class="weui-btn" type="primary" formType="submit">确定</button>
  </view>
</form>
```

3. 表单提交

表单提交是在表单验证通过后，将用户输入的信息保存到云数据库。

这里以实现“姓名”这个文本框的表单提交为例来介绍表单提交的实现方式，如何完整实现如图 15 所示的表单内容，请阅读本专栏源代码 `demo/pages/07_form_submit`。

首先我们需要在云数据库中新建一个数据库集合 `form_demo`，新建数据库集合的方法已在上一节中介绍。

然后我们需要编写提交表单数据到数据库的方法，将用户输入的姓名保存到云数据库，保存成功后提示表单提交成功，并显示存入云数据库的记录ID：

```
/**
 * 保存表单到云数据库
 */
saveForm: function(params) {
  const db = wx.cloud.database()
  //将用户输入的姓名保存到云数据库
  db.collection('form_demo').add({
      data: {
        //用户输入的姓名
        name: params.name,
        //使用云服务器时间记录表单提交时间
        createTime: db.serverDate()
      }
    })
    .then(res => {
      //提示表单提交成功，并显示存入云数据库的记录ID
      wx.showModal({
        title: '提交',
        content: '表单插入数据库成功, ID:' + res._id,
        showCancel: false,
        success(res) {
          wx.navigateBack({
            delta: 1,
          })
        }
      })
    })
}
```

最后将 2.4 中的：

```
  //如果表单验证成功，进行提示
  wx.showModal({
    title: '验证结果',
    content: '验证成功'
  })
```

修改为提交数据：

```
  //如果表单验证成功，提交数据到云数据库
  this.saveForm(params)
```

这样在点击提交按钮，通过表单验证后，将提交表单数据到云数据库。

提交表单数据到云数据库成功后的提示界面如图 19 所示。

图 19 表单提交成功界面

4. 云函数

在使用云开发进行小程序应用开发时，还有一个经常会用到的云开发能力-云函数。

有同学可能会问，到目前为止，我们编写的所有代码都是运行在小程序手机端的，而且看起来小程序手机端+云开发数据库就可以完成一个完整的小程序应用了，云函数有啥用？

云函数当然有用，而且有不可替代的作用。云函数的不可替代性在于它是运行在云开发的服务器端的。

怎么理解呢，我们来举一个例子。

我们现在要开发一个积分兑换实物商品的小程序，这个小程序要实现积分消费的功能。积分消费功能首先要查看用户的积分是不是够兑换他想兑换的商品，如果用户有足够的积分，就允许他兑换这个商品。

如果你把这个积分消费的功能逻辑写在小程序手机端，你可能瞬间变成百万“负”翁。

因为在小程序手机端你需要向云数据库请求三次数据，第一次是查询用户积分，如果用户积分大于等于兑换商品所需的积分就进行第二次数据库请求：插入一条用户兑换商品的记录，然后再进行第三次数据库请求：将用户的积分减去兑换商品消耗的积分。

这三次请求都是在手机端执行的，那么我就可以通过技术手段，跳过第一次和第三次数据请求，直接进行第二次数据库请求。也就是说，我可以不花一个积分，兑换所有的商品！

更惨的是，中国还有庞大的灰产行业链，如果一旦有人发现了这个BUG，几百万薅羊毛的能手，会像闻到血腥味的鲨鱼一样，疯狂利用这个漏洞免费兑换商品。

想想这个后果吧，你还觉得云函数没有用吗。

如果将这 3 次数据库请求都写在一个云函数中的，那么小程序手机端只能调用这个云函数来提交积分兑换请求。

云函数运行在服务器端，这个云函数的3次数据库请求顺序和程序逻辑，在小程序手机端是无法用技术手段篡改或跳过一部分只执行另一部分的。

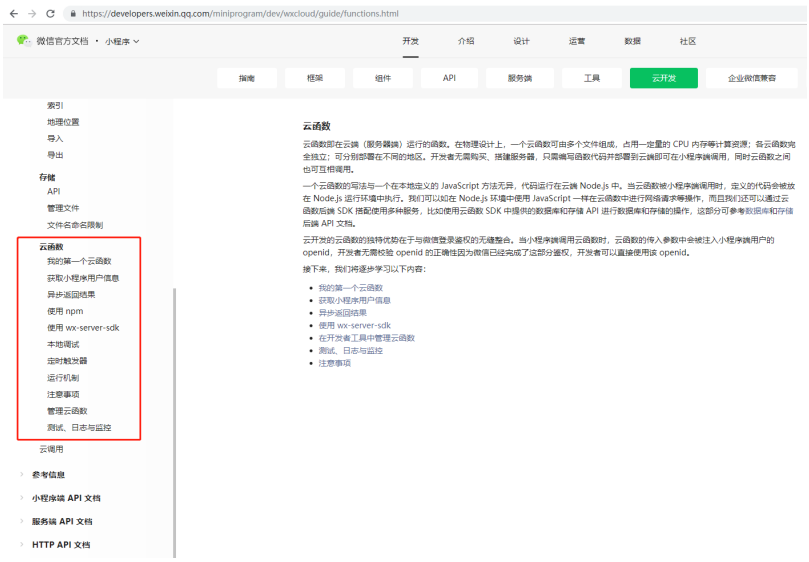
这样才能避免被羊毛党吸光血的惨案发生。

为了你的“钱”途，请认真学习和掌握云函数。

微信官方的“小程序开发文档”有详细的云函数教程，**云函数教程位置如下：**

- **入口网址：**云函数教程，**如果微信官方文档改版导致链接失效，请按图索骥。**
- **入口位置：在“小程序开发文档”的“云开发”栏目，如图 20 红框标出部分。**

图 20 云函数教程位置



5. 自动登录

云函数一个最基本也是最广泛的应用场景就是与微信登录鉴权的无缝整合，实现了微信自动登录。

不使用云开发，实现微信登录获取用户的 OpenId 需要一个复杂的流程，具体有多复杂可以参考“小程序开发文档”中的介绍：“指南”->“开放能力”->“小程序登录”。

而在云开发中使用云函数，只需要两行代码就可以实现：

```
const wxContext = cloud.getWXContext()
return {
  openid: wxContext.OPENID
}
```

云函数实现微信自动登录获取用户 OpenID 的 Demo 在本专栏源代码 `demo/pages/08_auto_login` 及 `cloudfunctions/autoLoginDemo` ，请结合微信官方的“小程序开发文档”中“获取小程序用户信息”一节阅读并理解云开发中获取用户 OpenID 的方法。

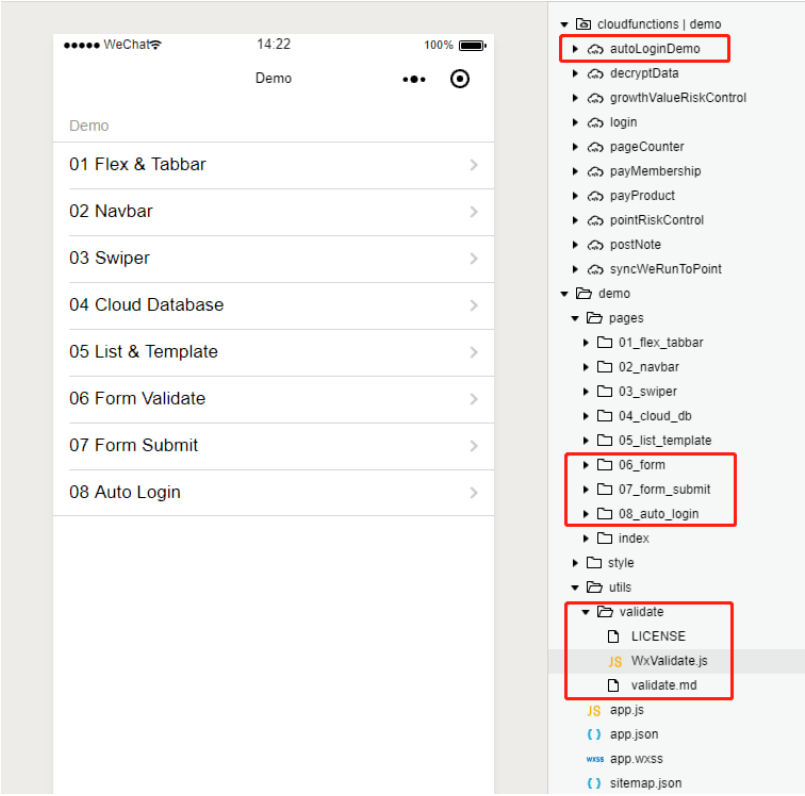
6. 专栏源代码

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 21 红框标出的位置。

图 21 本节源代码位置



本章小结

在本章，我们以“小程序开发文档”和“分类拆解法”为主线，向各位同学介绍了小程序开发的主要内容，包括页面布局、小程序组件、WeUI组件、云开发数据库创建与操作、表单、云函数。

只阅读本专栏的内容对入门小程序开发是远远不够的，微信官方的“小程序开发文档”中的内容需要反复阅读、实践、理解。

本章节已经给出了“小程序开发文档”基础内容的学习顺序，各位同学可以按照本章提到“小程序开发文档”的先后顺序来进行学习。

请各位同学在完全吸收了本章节内容，以及本章节提到的“小程序开发文档”内容，并完成实践环节后再开始后续章节学习。 如果不完成这些内容就直接阅读后续章节，很可能导致你出口成“脏”：MD，写的什么啊，根本读不懂。我们都不希望这种情况出现对吧，所以还是要辛苦你在这一章多花花心思了。

小程序是一个庞大的生态，没有任何一个人能记住小程序开发的所有细节。请不要背诵任何内容，通过不断实践自然会记住并运用越来越多的内容。**在实践中碰到问题，然后解决问题，是学习一本编程语言最快速的方式。**

所以，请将“小程序开发文档”作为你最重要的工具书，当你碰到任何问题，带着问题到“小程序开发文档”中寻找答案。

下节预告

下一节，我们将进入商业项目“会员制社交电商小程序”的编程实战环节，请做好准备：）。

实践环节

实践是通往大神之路的唯一捷径。

本节实操内容：

- 编写代码完成图 **15** 、图 **18**、 图 **19** 所示的页面，如碰到问题，请阅读本专栏源代码学习如何实现。
- 编写云函数，实现将用户 **OpenID** 显示到页面，如碰到问题，请阅读本专栏源代码学习如何实现。

08 数据交互：云开发让数据库操作如此简单

10 功能分析：3大商业小程序标配功能详解