# 微信小程序开发教程

1. 获取微信小程序的 AppID

如果你是收到邀请的开发者，我们会提供一个帐号，利用提供的帐号，登录 https://mp.weixin.qq.com ，就可以在网站的“设置”-“开发者设置”中，查看到微信小程序的 AppID 了，注意不可直接使用服务号或订阅号的 AppID 。

如果没有收到内测邀请，可以跳过本步骤



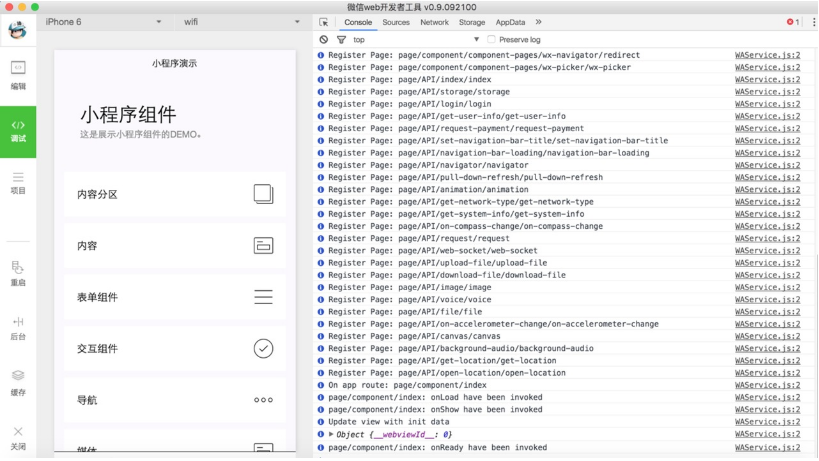
注意：如果我们不是用注册时绑定的管理员微信号，在手机上体验该小程序，那么我们还需要操作“绑定开发者”。即在“用户身份”-“开发者”模块，绑定上需要体验该小程序的微信号。本教程默认注册帐号、体验都是使用管理员微信号。

2. 创建项目

我们需要通过开发者工具，来完成小程序创建和代码编辑。

开发者工具安装完成后，打开并使用微信扫码登录。选择创建“项目”，填入上文获取到的 AppID ，设置一个本地项目的名称（非小程序名称），比如“我的第一个项目”，并选择一个本地的文件夹作为代码存储的目录，点击“新建项目”就可以了。

为方便初学者了解微信小程序的基本代码结构，在创建过程中，如果选择的本地文件夹是个空文件夹，开发者工具会提示，是否需要创建一个 quick start 项目。选择“是”，开发者工具会帮助我们在开发目录里生成一个简单的 demo。



项目创建成功后，我们就可以点击该项目，进入并看到完整的开发者工具界面，点击左侧导航，在“编辑”里可以查看和编辑我们的代码，在“调试”里可以测试代码并模拟小程序在微信客户端效果，在“项目”里可以发送到手机里预览实际效果。

3. 编写代码

创建小程序实例

点击开发者工具左侧导航的“编辑”，我们可以看到这个项目，已经初始化并包含了一些简单的代码文件。最关键也是必不可少的，是 app.js、app.json、app.wxss 这三个。其中，.js后缀的是脚本文件，.json后缀的文件是配置文件，.wxss后缀的是样式表文件。微信小程序会读取这些文件，并生成小程序实例。

下面我们简单了解这三个文件的功能，方便修改以及从头开发自己的微信小程序。

app.js是小程序的脚本代码。我们可以在这个文件中监听并处理小程序的生命周期函数、声明全局变量。调用框架提供的丰富的 API，如本例的同步存储及同步读取本地数据。想了解更多可用 API，可参考 API 文档

//app.jsApp({  onLaunch: function () {    //调用API从本地缓存中获取数据    var logs = wx.getStorageSync('logs') || []    logs.unshift(Date.now())    wx.setStorageSync('logs', logs)  },  getUserInfo:function(cb){    var that = this;    if(this.globalData.userInfo){      typeof cb == "function" && cb(this.globalData.userInfo)    }else{      //调用登录接口      wx.login({        success: function () {          wx.getUserInfo({            success: function (res) {              that.globalData.userInfo = res.userInfo;              typeof cb == "function" && cb(that.globalData.userInfo)            }          })        }      });    }  },  globalData:{    userInfo:null  } })

app.json 是对整个小程序的全局配置。我们可以在这个文件中配置小程序是由哪些页面组成，配置小程序的窗口背景色，配置导航条样式，配置默认标题。注意该文件不可添加任何注释。更多可配置项可参考配置详解

{  "pages":[    "pages/index/index",    "pages/logs/logs"  ],  "window":{    "backgroundTextStyle":"light",    "navigationBarBackgroundColor": "#fff",    "navigationBarTitleText": "WeChat",    "navigationBarTextStyle":"black"  } }

app.wxss 是整个小程序的公共样式表。我们可以在页面组件的 class 属性上直接使用 app.wxss 中声明的样式规则。

/\*\*app.wxss\*\*/.container {  height: 100%;  display: flex;  flex-direction: column;  align-items: center;  justify-content: space-between;  padding: 200rpx 0;  box-sizing: border-box; }

创建页面

在这个教程里，我们有两个页面，index 页面和 logs 页面，即欢迎页和小程序启动日志的展示页，他们都在 pages 目录下。微信小程序中的每一个页面的【路径+页面名】都需要写在 app.json 的 pages 中，且 pages 中的第一个页面是小程序的首页。

每一个小程序页面是由同路径下同名的四个不同后缀文件的组成，如：index.js、index.wxml、index.wxss、index.json。.js后缀的文件是脚本文件，.json后缀的文件是配置文件，.wxss后缀的是样式表文件，.wxml后缀的文件是页面结构文件。

index.wxml 是页面的结构文件：

<!--index.wxml--><view class="container">  <view  bindtap="bindViewTap" class="userinfo">    <image class="userinfo-avatar" src="{{userInfo.avatarUrl}}" background-size="cover"></image>    <text class="userinfo-nickname">{{userInfo.nickName}}</text>  </view>  <view class="usermotto">    <text class="user-motto">{{motto}}</text>  </view></view>

本例中使用了<view/>、<image/>、<text/>来搭建页面结构，绑定数据和交互处理函数。

index.js 是页面的脚本文件，在这个文件中我们可以监听并处理页面的生命周期函数、获取小程序实例，声明并处理数据，响应页面交互事件等。

//index.js//获取应用实例var app = getApp() Page({  data: {    motto: 'Hello World',    userInfo: {}  },  //事件处理函数  bindViewTap: function() {    wx.navigateTo({      url: '../logs/logs'    })  },  onLoad: function () {    console.log('onLoad')    var that = this    //调用应用实例的方法获取全局数据    app.getUserInfo(function(userInfo){      //更新数据      that.setData({        userInfo:userInfo      })    })  } })

index.wxss 是页面的样式表：

/\*\*index.wxss\*\*/.userinfo {  display: flex;  flex-direction: column;  align-items: center; }.userinfo-avatar {  width: 128rpx;  height: 128rpx;  margin: 20rpx;  border-radius: 50%; }.userinfo-nickname {  color: #aaa; }.usermotto {  margin-top: 200px; }

页面的样式表是非必要的。当有页面样式表时，页面的样式表中的样式规则会层叠覆盖 app.wxss 中的样式规则。如果不指定页面的样式表，也可以在页面的结构文件中直接使用 app.wxss 中指定的样式规则。

index.json 是页面的配置文件：

页面的配置文件是非必要的。当有页面的配置文件时，配置项在该页面会覆盖 app.json 的 window 中相同的配置项。如果没有指定的页面配置文件，则在该页面直接使用 app.json 中的默认配置。

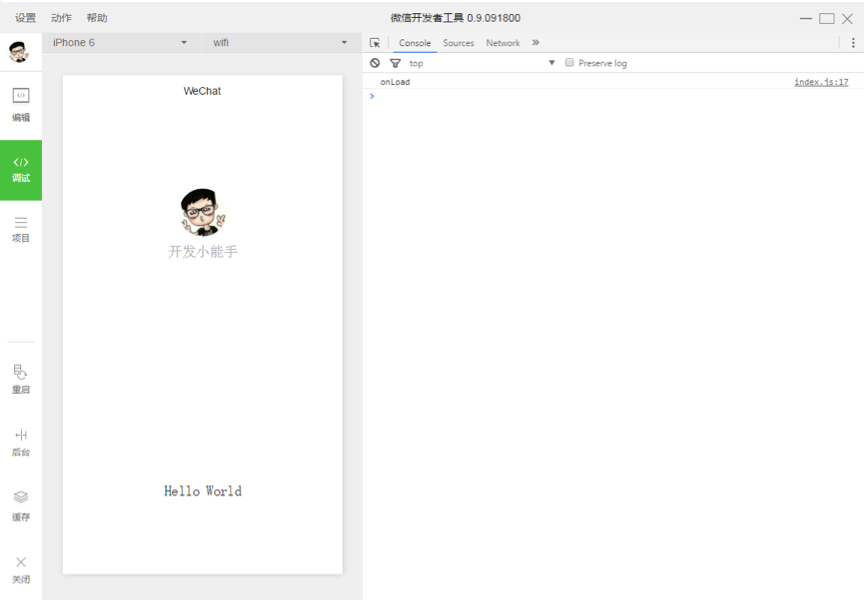
logs 的页面结构

<!--logs.wxml--><view class="container log-list">  <block wx:for="{{logs}}" wx:for-item="log">    <text class="log-item">{{index + 1}}. {{log}}</text>  </block></view>

logs 页面使用 <block/> 控制标签来组织代码，在 <block/> 上使用 wx:for 绑定 logs 数据，并将 logs 数据循环展开节点

//logs.jsvar util = require('../../utils/util.js') Page({  data: {    logs: []  },  onLoad: function () {    this.setData({      logs: (wx.getStorageSync('logs') || []).map(function (log) {        return util.formatTime(new Date(log))      })    })  } })

运行结果如下：



4. 手机预览

开发者工具左侧菜单栏选择"项目"，点击"预览"，扫码后即可在微信客户端中体验。

