第5章

小程序项目的配置与生命周期

本章学习目标

- 进一步熟悉小程序的全局配置，掌握在 app.json 文件中进行页面、窗口、tabBar 等的配置。
- 围绕着 app.js 文件，理解小程序的生命周期，熟悉 App 对象的使用。
- 了解小程序页面配置与小程序全局配置的区别，并围绕页面的 JS 文件学习页面生命周期过程。
- 学习页面间如何进行参数传递。

5.1 app.json 配置属性

app.json 为全局配置文件，主要用于项目配置，包含诸多属性。实际开发过程中也可参考微信官方文档，选择“框架”→“小程序配置”→“全局配置”命令进行查看。app.json 配置属性的介绍如表 5-1 所示。

表 5-1 app.json 配置属性

属　性	类　型	描　述
pages	string[]	必填，记录小程序所有页面的路径地址
window	object	可选，用于设置页面的窗口表现，例如导航栏的背景颜色、标题文字内容以及文字颜色等
tabBar	object	可选，用于设置页面底部 Tab 工具条的表现
networkTimeout	object	可选，用于设置各种网络请求的超时时间
navigateToMiniProgramAppIDList	string[]	可选，需要跳转的小程序列表，最多允许填写 10 个
functionalPages	boolean	可选，是否启用插件功能页，默认关闭。小程序运行的基础库最低版本为 2.1.0
debug	boolean	可选，用于设置是否开启调试模式
subpackages	oobject[]	可选，分包结构配置。小程序运行的基础库最低版本为 1.7.3

续表

属　性	类　型	描　述
preloadRule	object	分包预下载规则。小程序运行的基础库最低版本为 2.3.0
workers	string	可选，worker 是运行在后台的 JavaScript，独立于其他脚本，不会影响页面的性能。该属性指出 worker 代码放置的目录，小程序运行的基础库最低版本为 1.9.90
requiredBackgroundModes	string[]	需要在后台使用的能力，如音乐播放
plugins	object	使用到的插件，小程序运行的基础库版本为 1.9.6
sitemapLocation	string	必需，sitemap.json 全局配置文件，作用是用户搜索小程序内部的某个页面的相关设置。这里的该属性用于定位该文件位置
resizable	boolean	可选，iPad 小程序是否支持屏幕旋转，默认关闭。小程序运行的基础库版本为 2.3.0
style	string	指定使用升级后的 WeUI 样式

小程序的运行要求与微信客户端版本相匹配，每一个基础库只能在对应的客户端版本上运行，高版本的基础库无法兼容低版本的微信客户端。基础库版本可在微信开发工具中自定义，具体运行情况以真机调试为准，为了避免 Android 与 iOS 系统运行表现不同，建议同时使用两种不同系统的机型进行小程序的调试。

5.2　页面配置

App 的页面配置指的是 pages 属性，pages 数组的第一个页面将默认作为小程序的启动页。利用开发工具新建页面时，pages 属性对应的数组将自动添加该页面的路径，若是在硬盘中添加文件的形式则不会触发该效果。pages 属性配置示例代码如下所示：

```
1.  {
2.    "pages": [
3.      "pages/index/index",
4.      "pages/logs/logs"
5.    ],
6.    }
```

上述代码中 pages 数组中页面路径之间用英文逗号分开，最后一个页面路径不加逗号。这里要注意 JSON 文件的语法，如不能添加注释在 JSON 文件中等。

5.3　窗口配置

window 属性对应的是一个 json 对象，可用于配置小程序顶部 navigationBar 的颜色、标题文字，且均作用于全局。如果页面的 JSON 文件不再进行单独的配置，全部页面都将默认使用 app.json 文件中 window 属性的配置。window 的子属性说明如表 5-2 所示。

表 5-2 window 属性说明

属 性	类型	默认值	解 释
navigationBar backgroundColor	hexcolor	#000000	导航栏背景颜色，默认值表示黑色
navigationBarTextStyle	string	white	导航栏标题颜色，默认值表示白色，该属性值只能是 white 或 black
navigationBarTitleText	string		导航栏标题文字内容，默认无文字内容
navigationStyle	string	default	导航栏样式，default 表示默认格式。custom 表示自定义导航栏，只保留右上角的小图标（微信版本 6.6.0 以上支持此功能）
backgroundColor	hexcolor	#ffffff	窗口的背景颜色，默认值表示白色
backgroundTextStyle	string	dark	下拉加载的样式，选填 dark 或 light
backgroundColorTop	string	#ffffff	顶部窗口的背景颜色，只有 iOS 有效（微信版本 6.5.16 以上支持此功能）
backgroundColorBottom	string	#ffffff	底部窗口的背景颜色，只有 iOS 有效（微信版本 6.5.16 以上支持此功能）
enablePullDownRefresh	boolean	false	是否开启下拉刷新功能
onReachBottomDistance	number	50	页面上拉触底事件触发时距页面底部距离，单位为像素（px）

5.4 tabBar 配置

5.4.1 tabBar 属性

tabBar 是固定在软件主界面底部的类似于单选按钮作用的横条，选中不同的选项按钮后，主界面就切换成不同的页面。tabBar 的属性配置如表 5-3 所示。

表 5-3 tabBar 的属性配置

属 性	类型	是否必填	默认	描 述
color	hexcolor	是		tab 上的文字默认颜色
selectedColor	hexcolor	是		tab 上的文字选中时的颜色
backgroundColor	hexcolor	是		tab 的背景色
borderStyle	string	否	black	tabBar 边框颜色，仅支持 black/white
list	array	是		tab 的列表，包含 4 个子属性：页面的路径、默认图标路径、选中图标路径、文字
position	string	否	bottom	tabBar 的位置，仅支持 bottom/top

表 5-3 中 list 子属性对应的参数为 pagePath、iconPath、selectedIconPath 和 text。其中关于 pagePath 属性的路径需要在 app.json 的 pages 中先定义，而 iconPath（tabBar 的图标路径）放置的图片大小限制为 40KB，建议大小为 81px×81px，不支持网络图片（即 URL）。selectedIconPath 与 iconPath 的限制相同。但是当表中的 position 属性值设为 top 时，不显示图标。

5.4.2 tabBar 配置示例

1.3.3 节中介绍的“微信小程序示例”主界面使用了 tabBar，如图 5-1 所示。

(a) 程序底部的tabBar　　(b) tabBar的切换

图 5-1　官方示例程序 tabBar 配置示例图

“微信小程序示例”的 app.json 源代码如下：

```
"tabBar": {
    "color": "#7A7E83",
    "selectedColor": "#3cc51f",
    "borderStyle": "black",
    "backgroundColor": "#ffffff",
    "list": [{ "pagePath": "page/component/index",
            "iconPath": "image/icon_component.png",
            "selectedIconPath": "image/icon_component_HL.png",
            "text": "组件" },
           { "pagePath": "page/API/index",
            "iconPath": "image/icon_API.png",
            "selectedIconPath": "image/icon_API_HL.png",
            "text": "接口" },
            { "pagePath": "page/cloud/index",
            "iconPath": "image/icon_cloud.png",
            "selectedIconPath": "image/icon_cloud_HL.png",
            "text": "云开发" } ]
  }
```

5.5 网络超时配置

networkTimeout 属性用于设置网络超时时间，其子属性配置如表 5-4 所示。

表 5-4 networkTimeout 的子属性配置

属性	类型	是否必填	默认值	说　明
request	number	否	60000	wx.request 超时时间，单位为毫秒
connectSocket	number	否	60000	wx.connectSocket 超时时间，单位为毫秒
uploadFile	number	否	60000	wx.uploadFile 超时时间，单位为毫秒
downloadFile	number	否	60000	wx.downloadFile 超时时间，单位为毫秒

5.6 权限配置

5.6.1 接口权限

1. 代码说明

permission 属性用于小程序接口权限相关设置，字段类型为 object。以位置权限为例，小程序若想要获取用户的位置信息，需在 app.json 中配置 permission 属性，代码如下所示：

```
{
  "pages": ["pages/index/index"],
  "permission": {
    "scope.userLocation": {
      "desc": "你的位置信息将用于小程序位置接口的效果展示"
    }
  }
}
```

用户信息、地理位置等权限出于隐私原因需经过用户授权后才能获取，因此，在首次调用小程序的 wx.getLocation 等接口获取用户定位时会跳出弹窗，提醒用户进行授权，代码如下所示：

```
wx.getSetting({ //wx.getSetting 用于获取小程序当前需授权接口的授权情况
      success(res) {
        if (!res.authSetting['scope.userLocation']) {// scope 是一个长列表，包含所有权限
          wx.authorize({                          //调用 wx.authorize 弹出弹窗
            scope: 'scope.userLocation',
            success() {
              //用户已经同意小程序使用定位功能，后续调用不会弹出弹窗询问
              // wx.getLocation({...})
            }
          })
```

```
11.            }
12.        }
13.    })
```

上述代码共涉及 3 个 API 函数。第 1 个是 wx.getSetting，用于获取小程序当前需授权接口的授权情况，调用成功后在 success 回调函数中接收该 API 函数返回的结果。在第 3 句代码中做判断，若 scope.userLocation 即位置权限未经用户授权，则会调用第 2 个 API 函数 wx.authorize 提示用户进行授权，这时弹窗显示的位置用途即 permission 配置内容。用户点击“同意”按钮后，再调用第 3 个 API 函数 wx.getLocation 获取用户的地理位置，且后续调用都不会再进行弹窗询问。

2. 运行效果

permission 属性配置地理位置用途的授权弹窗如图 5-2 所示。

图 5-2　获取位置权限提示

需要注意的是，图 5-2 的项目是使用了申请正式 AppID 的小程序项目，若使用测试号，如果测试号小程序本身已授权位置权限，则上述授权代码不会再弹出弹窗询问授权。

5.6.2　后台能力权限

requiredBackgroundModes 属性用于申明需要后台运行的能力，属性值类型为数组。目前该属性支持以下后台能力。

- audio：后台音乐播放。
- location：后台定位。

示例代码如下：

```
{
  "pages": ["pages/index/index"],
  "requiredBackgroundModes": ["audio", "location"]
}
```

在此处申明了后台运行的接口，开发版和体验版上可以直接生效，正式版还需要通过审核。


视频讲解

5.7　小程序的生命周期

5.7.1　小程序生命周期函数

小程序的生命周期主要有初始化、启动和切后台，分别对应 onLaunch、onShow、onHide 3 个函数，而这 3 个函数是定义在 app.js 项目逻辑文件中，在注册小程序时作为参数传入函数 App()的。小程序生命周期函数相关说明如表 5-5 所示。

表 5-5　小程序生命周期函数相关说明

属　性	类　型	是否必填	说　明
onLaunch	function	否	生命周期回调：监听小程序初始化
onShow	function	否	生命周期回调：监听小程序启动或切前台
onHide	function	否	生命周期回调：监听小程序切后台

5.7.2　小程序生命周期测试案例

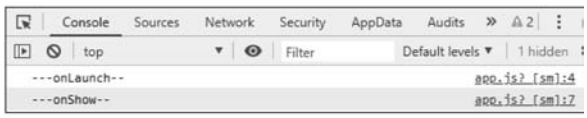
1. 运行效果

下面实现小程序项目的生命周期测试案例，其中包含 3 个操作，首先是启动项目时触发的两个生命周期函数，分别是 onLaunch 和 onShow。其次是小程序切后台时会触发 onHide 函数。最后再次打开小程序时，小程序并不会重新加载，而是只触发 onShow 函数。案例的具体操作步骤与演示效果如图 5-3 所示。

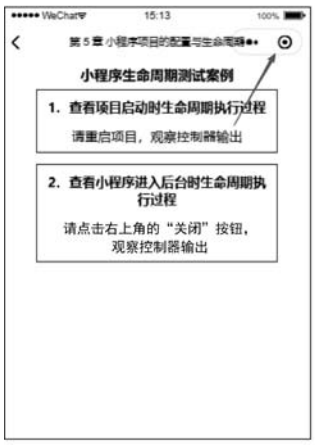
图 5-3(f)中出现的场景是指用户在进入该小程序时是以何种方式进入的，对每一种进入方式进行了编号，称为场景值。常见的进入小程序的方式有扫描小程序码、通过微信客户端的发现栏等。若开发者需要获取用户进入该小程序的方式，可以在 App 的 onLaunch 和 onShow 中通过参数获取，也可以通过调用 API 函数 wx.getLaunchOptionsSync 获取上述场景值。

随机选中一个场景值再次进入小程序，控制栏会只打印“---onShow--”，可见小程序并没有被真正关闭而是进入后台运行，因此不会再进行初始化操作。前台的意思是小程序页面正处于当前手机顶部正在运行，而后台的意思是小程序页面被其他页面打断，如突然有电话接入的情况。在小程序实际运行过程中，很可能出现被其他页面所打断的情况，这时需要在 onHide 中执行把相关操作暂停的逻辑，而在 onShow 中继续操作，这点对于游戏的开发尤为重要。

(a) 启动项目

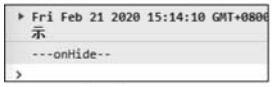


(b) 控制器输出

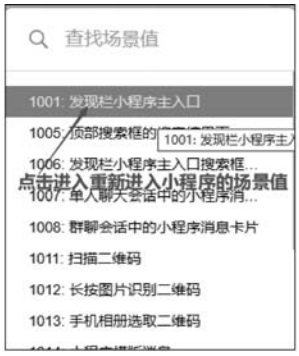


(c) 点击右上角的“关闭”按钮

(d) 小程序进入后台



(e) 触发onHide



(f) 选择某项重新进入方式

(g) 触发onShow

图 5-3　小程序生命周期测试案例的具体操作步骤与演示效果

2. 代码说明

新建项目 Chapter05，新建 lifeCycle 页面。在自动新建的 app.js 文件中可以看到如下代码：

```
1.  //项目逻辑文件 app.js
2.  App({
3.    onLaunch: function () {
4.      //省略
5.    },
6.    globalData: { userInfo: null }
7.  })
```

如 1.4 节所述，项目逻辑文件 app.js 中 App 函数用于进行小程序的注册，而 App 函数中传入的大型参数中就包括 onLaunch 方法，它主管整个项目生命周期的初始化环节，除了 onLaunch 方法之外，还有 onShow 以及 onHide，可以添加上这两个方法在 app.js 中进行测试，代码如下所示：

```
1.  //项目逻辑文件 app.js
2.  App({
3.    onLaunch: function () {
4.      console.log("--- onLaunch --");
5.    },
6.    onShow: function () {
7.      console.log("--- onShow --");
8.    },
9.    onHide: function () {
10.     console.log("--- onHide --");
11.   }
12. })
```

5.8 使用 app 对象的案例讲解

视频讲解

5.8.1 实现效果

调用 app 对象可以使用 app.js 中定义的全局函数和变量。本节介绍一个传递用户名的案例，首先定义 app.js 中的变量和函数，然后在其他页面中使用 app 对象，并调用 app.js 中的函数以获取全局用户名，案例实现效果如图 5-4 所示。

图 5-4(c)所示的用户名是由 app.globalData.userName 赋值的，这里的 app 就是在 JS 页面中获取到的 app 对象。app.globalData.userName 值在 app.js 中被 testApp 函数赋值为“Toky”，在 5.8.2 节的代码说明中可以看到。

(a) 页面初始状态　(b) 点击“点击调用”按钮　(c) 成功获取全局用户名

图 5-4　利用 app 对象调用 app.js 的全局变量与函数

5.8.2　在 app.js 中定义全局变量与函数

如前所述，在 app.js 里定义全局变量与函数，代码如下所示：

```
1.  //app.js
2.  App({
3.    onLaunch: function () {
4.      //项目启动时逻辑
5.    },
6.    /** 定义在 app.js 中的全局函数 */
7.    testApp(){
8.      this.globalData.userName = 'Toky'
9.    },
10.   globalData: {
11.     userName: null //定义在 app.js 中的全局变量
12.   }
13. })
```

5.8.3　页面获取 app 对象

1. WXML 代码

```
1.  <!-- index.wxml -->
2.  <view class = 'container'>
3.    <view class = 'page - body'>
4.      <text class = 'h1'>使用 app 对象的案例讲解</text>
5.      <view class = 'demo - box'>
6.        <view class = "title">点击按钮调用 app.js 中 testApp 函数</view>
7.        <button type = "primary" bindtap = "getUserName">点击调用</button>
8.        <view class = "title">app.js 中定义的全局用户名是:
9.          <text>{{name}}</text>
```

```
        </view>
      </view>
    </view>
</view>
```

2. JS 逻辑代码

index.js 代码如下：

```
//index.js
const app = getApp()          //获取应用实例
Page({
  data: {
    name:null
  },
  //事件处理函数
  getUserName: function() {
    app.testApp()
    this.setData({ name: app.globalData.userName})
  }
})
```

5.9　本章小结

本章主要讲解了小程序项目层面的配置与生命周期等，具体涉及 app.json 文件作用的小程序全局配置、app.js 文件作用的小程序项目的 3 个周期函数运行情况，最后结合案例讲解了 app 对象的具体使用。