新手入门选修教程：小程序的框架及逻辑层

一：小程序的框架-mina

MINA

MINA(MINA IS NOT APP) 是在微信中开发**小程序**的框架。

MINA 的目标是通过尽可能简单、高效的方式让开发者可以在微信中开发具有原生 APP 体验的服务。

MINA 提供了自己的视图层描述语言 WXML 和 WXSS，以及基于 JavaScript 的逻辑层框架，并在视图层与逻辑层间提供了数据传输和事件系统，可以让开发者可以方便的聚焦于数据与逻辑上。

文件结构

框架程序包含一个描述整体程序的 app 和多个描述各自页面的 page。

一个框架程序主体部分由三个文件组成，必须放在项目的根目录，如下：

文件	必填	作用
app.js	是	小程序逻辑
app.json	是	小程序公共设置
app.wxss	否	小程序公共样式表

一个框架页面由四个文件组成，分别是：

文件类型	必填	作用
js	是	页面逻辑
wxml	是	页面结构
wxss	否	页面样式表
json	否	页面配置

配置

我们使用 app.json 文件来对微信小程序进行全局配置，决定页面文件的路径、窗口表现、设置网络超时时间、设置多 tab 等。

以下是一个包含了所有配置选项的简单配置 app.json ：

```
{
"pages": [
"pages/wechat/wechat",
"pages/note/note",
"pages/find/find",
"pages/mine/mine",
```

```
"pages/message/message",
"pages/audio/audio",
"pages/info/info",
"pages/newfriend/newfriend",
"pages/moments/moments",
"pages/picker/picker",
"pages/upload/upload",
"pages/ws/ws",
"pages/index/index",
"pages/logs/logs"
],
"window":{
"navigationBarBackgroundColor": "#000",
"navigationBarTextStyle": "white",
"navigationBarTitleText": "",
"backgroundColor": "#eee",
```

```
"backgroundTextStyle": "dark"

},

"tabBar": {

"backgroundColor":"#333",

"selectedColor":"red",

"list": [{

"pagePath": "pages/wechat/wechat",

"iconPath":"image/wechat.png",

"selectedIconPath":"image/wechat_2.png",

"text": "微信"

}, {

"pagePath": "pages/note/note",

"iconPath":"image/note.png",

"selectedIconPath":"image/note_2.png",

"text": "通讯录"

}, {
```

```
"pagePath": "pages/find/find",
"iconPath":"image/find.png",
"selectedIconPath":"image/find_2.png",
"text": "发现"
}, {
"pagePath": "pages/mine/mine",
"iconPath":"image/mine.png",
"selectedIconPath":"image/mine_2.png",
"text": "我"
}]
},
"networkTimeout": {
"request": 10000,
"downloadFile": 10000
},
"debug": true
```

```
}
```

app.json 配置项列表

属性	类型	必填	描述
pages	Array	是	设置页面路径
window	Object	否	设置默认页面的窗口表现
tabBar	Object	否	设置底部 tab 的表现
networkTimeout	Object	否	设置网络超时时间
debug	Boolean	否	设置是否开启 debug 模式

pages

接受一个数组，每一项都是字符串，来指定小程序由哪些页面组成。每一项代表对应页面的【路径+文件名】信息，**数组的第一项代表小程序的初始页面**。小程序中新增/减少页面，都需要对 pages 数组进行修改。

文件名不需要写文件后缀，因为框架会自动去寻找路径.json,.js,.wxml,.wxss 的四个文件进行整合。

window

用于设置小程序的状态栏、导航条、标题、窗口背景色。

属性	类型	默认值	描述
navigationBarBackgroundColor	HexColor	#000000	导航栏背景颜色，如"#000000"

navigationBarTextStyle	String	white	导航栏标题颜色，仅支持 black/white
navigationBarTitleText	String		导航栏标题文字内容
backgroundColor	HexColor	#ffffff	下拉窗口的背景色
backgroundTextStyle	String	dark	下拉背景字体、loading 图的样式，仅支持 dark/light

tabBar

如果我们的小程序是一个多 tab 应用（客户端窗口的底部有 tab 栏可以切换页面），那么我们可以通过 tabBar 配置项指定 tab 栏的表现，以及 tab 切换时显示的对应页面。

tabBar 是一个数组，**只能配置最少 2 个、最多 5 个 tab**，tab 按数组的顺序排序。

属性说明：

属性	类型	必填	默认值	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 上边框的颜色， 仅支持 black/white
list	Array	是		tab 的列表，详见 list 属性说明，最少 2 个、最多 5 个 tab

其中 list 接受一个数组，数组中的每个项都是一个对象，其属性值如下：

属性	类型	必填	说明
pagePath	String	是	页面路径，必须在 pages 中先定义
text	String	是	tab 上按钮文字
iconPath	String	是	图片路径，icon 大小限制为 40kb
selectedIconPath	String	是	选中时的图片路径，icon 大小限制为 40kb

networkTimeout

可以设置各种网络请求的超时时间。

属性说明：

属性	类型	必填	说明

request	Number	否	wx.request 的超时时间，单位毫秒
connectSocket	Number	否	wx.connectSocket 的超时时间，单位毫秒
uploadFile	Number	否	wx.uploadFile 的超时时间，单位毫秒
downloadFile	Number	否	wx.downloadFile 的超时时间，单位毫秒

debug

可以在开发者工具中开启 debug 模式，在开发者工具的控制台面板，调试信息以 info 的形式给出，其信息有 Page 的注册，页面路由，数据更新，事件触发 。 可以帮助开发者快速定位一些常见的问题。

每个页面的配置文件（.json）

每一个小程序页面也可以使用.json 文件来对本页面的窗口表现进行配置。 页面的配置比 app.json 全局配置简单得多，只是设置 app.json 中的 window 配置项的内容，页面中配置项会覆盖 app.json 的 window 中相同的配置项。

页面的.json 只能设置 window 相关的配置项，以决定本页面的窗口表现，所以无需写 window 这个键，如：

```
{
```

```
"navigationBarBackgroundColor":
"#ffffff",
"navigationBarTextStyle":
"black",
"navigationBarTitleText":
"微信接口功能演示",
"backgroundColor":
"#eeeeee",
"backgroundTextStyle":
"light"
}
```

二：逻辑层

App()

App()函数用来注册一个小程序。接受一个 object 参数，其指定小程序的生命周期函数等。

object 参数说明：

属性	类型	描述	触发时机
onLaunch	Function	生命周期函数--监听小程序初始化	当小程序初始化完成时，会触发 onLaunch（全局只触发一次）
onShow	Function	生命周期函数--监听小程序显示	当小程序启动，或从后台进入前台显示，会触发 onShow
onHide	Function	生命周期函数--监听小程序隐藏	当小程序从前台进入后台，会触发 onHide
其他	Any	开发者可以添加任意的函数或数据到 Object 参数中，用 this 可以访问	

前台、后台定义： 当用户点击左上角关闭，或者按了设备 Home 键离开微信，小程序并没有正在的销毁，而是进入了后台；

当再次启动微信或再次打开小程序，又会从后台进入前台。

只有当小程序进入后台一定时间，或者系统资源占用过高，才会被真正的销毁。

[javascript] view plain copy

```
1.  //app.js
2.  App({
3.    onLaunch: function () {
4.      //调用 API 从本地缓存中获取数据
5.      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,
    ceshi:"I am global data"
  }
})
```

getApp()

我们提供了全局的 `getApp()`函数，可以获取到小程序实例。

[javascript] view plain copy

```
// other.js
var appInstance = getApp()
console.log(appInstance.globalData) // I am global data
```

注意：

App()必须在 app.js 中注册，且不能注册多个。

不要在定义于 App()内的函数中调用 getApp()，使用 this 就可以拿到 app 实例。

不要在 onLaunch 的时候调用 getCurrentPage()，此时 page 还没有生成。

通过 getApp 获取实例之后，不要私自调用生命周期函数。

Page

Page()函数用来注册一个页面。接受一个 object 参数，其指定页面的初始数据、生命周期函数、事件处理函数等。

object 参数说明：

属性	类型	描述
data	Object	页面的初始数据
onLoad	Function	生命周期函数--监听页面加载
onReady	Function	生命周期函数--监听页面渲染完成
onShow	Function	生命周期函数--监听页面显示

onHide	Function	生命周期函数--监听页面隐藏
onUnload	Function	生命周期函数--监听页面卸载
其他	Any	开发者可以添加任意的函数或数据到 Object 参数中，用 this 可以访问

初始化数据

初始化数据将作为页面的第一次渲染。data 将会以 JSON 的形式由逻辑层传至渲染层，所以其数据必须是可以转成 JSON 的格式：字符串，数字，布尔值，对象，数组。

渲染层可以通过 WXML 对数据进行绑定。

示例代码：

[javascript] view plain copy

```
<view>{{text}}</view>
<view>{{array[0].msg}}</view>
Page({
  data: {
    text: 'init data',
    array: [{msg: '1'}, {msg: '2'}]
  }
})
```

事件处理函数

？除了初始化数据和生命周期函数，Page 中还可以定义一些特殊的函数：事件处理函数。在渲染层可以在组件中加入事件绑定，当达到触发事件时，就会执行 Page 中定义的事件处理函数。

示例代码：

[javascript] view plain copy

```
<view bindtap="viewTap"> click me </view>
Page({
  viewTap: function() {
    console.log('view tap')
  }
})
```

Page.prototype.setData()

setData 函数用于将数据从逻辑层发送到视图层，同时改变对应的 this.data 的值。

注意：

1. **直接修改 this.data 无效，无法改变页面的状态，还会造成数据不一致。**
2. **单次设置的数据不能超过 1024kB，请尽量避免一次设置过多的数据。**

setData()参数格式

接受一个对象，以 key，value 的形式表示将 this.data 中的 key 对应的值改变成 value。

其中 key 可以非常灵活，以数据路径的形式给出，如 array[2].message，a.b.c.d，并且不需要在 this.data 中预先定义。

示例代码：

[javascript] view plain copy

```
<!--index.wxml-->
<view>{{text}}</view>
<button bindtap="changeText"> Change normal data </button>
<view>{{array[0].text}}</view>
<button bindtap="changeItemInArray"> Change Array data </button>
<view>{{obj.text}}</view>
<button bindtap="changeItemInObject"> Change Object data </button>
<view>{{newField.text}}</view>
<button bindtap="addNewField"> Add new data </button>
//index.js
Page({
  data: {
    text: 'init data',
    array: [{text: 'init data'}],
    object: {
      text: 'init data'
    }
  },
  changeText: function() {
    // this.data.text = 'changed data'  // bad, it can not work
    this.setData({
      text: 'changed data'
    })
  },
  changeItemInArray: function() {
    // you can use this way to modify a danamic data path
    var changedData = {}
    var index = 0
    changedData['array[' + index + '].text'] = 'changed data'
```

```
    this.setData(changedData)
  },
  changeItemInObject: function(){
    this.setData({
      'object.text': 'changed data'
    });
  },
  addNewField: function() {
    this.setData({
      'newField.text': 'new data'
    })
  }
})
```

模块化

我们可以将一些公共的代码抽离成为一个单独的 js 文件，作为一个模块。模块只有通过 module.exports 才能对外暴露接口。

[javascript] view plain copy

```
// common.js
function sayHello(name) {
  console.log('Hello ' + name + '!')
}
module.exports = {
  sayHello: sayHello
}
```

?在需要使用这些模块的文件中，使用 require(path)将公共代码引入。

[javascript] view plain copy

```
var common = require('common.js')
Page({
  helloMINA: function() {
    common.sayHello('MINA')
  }
})
```