微信小程序 - 使用 uni-app 开发小程序以及部分功能实现

文章目录

一、uni-app

官网：https://uniapp.dcloud.io/

1、简介

uni-app 是一个使用 Vue.js (opens new window) 开发所有前端应用的框架，开发者编写一套代码，可发布到iOS、Android、Web（响应式）、以及各种小程序（微信/支付宝/百度/头条/飞书/QQ/快手/钉钉/淘宝）、快应用等多个平台；

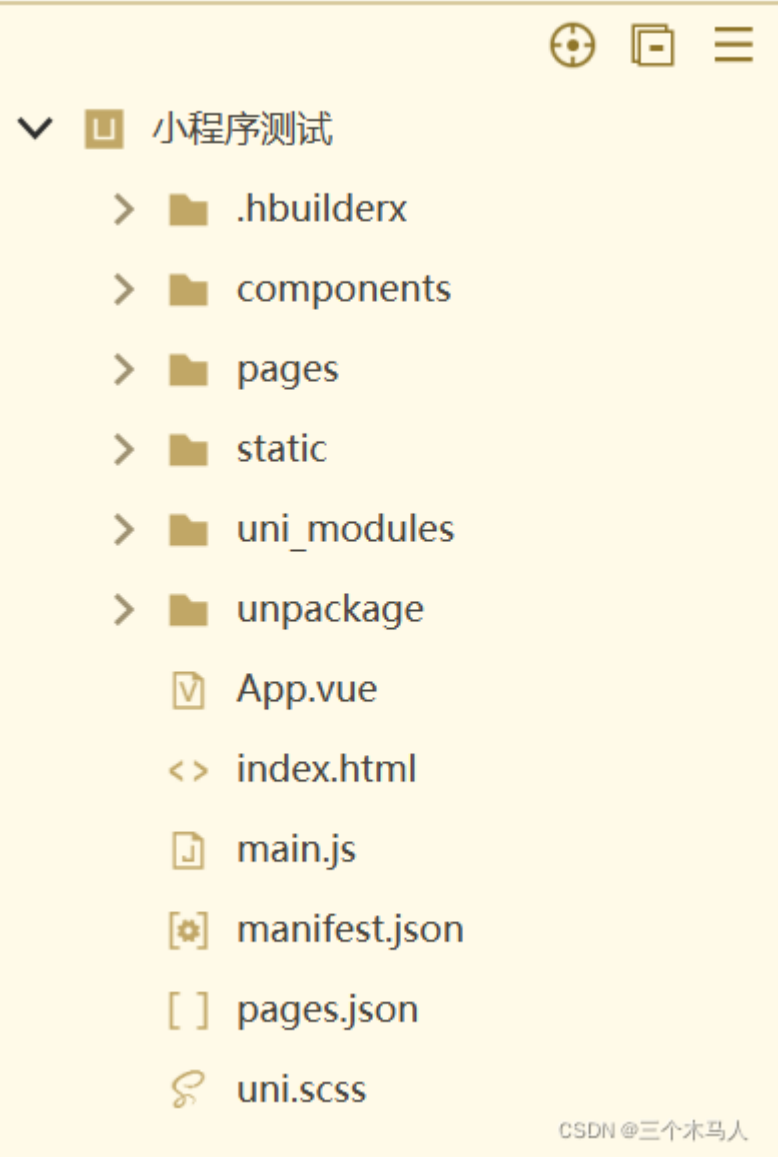
2、开发工具

uni-app 推荐使用 Hbuilderx 开发工具来开发项目，下载地址：https://www.dcloud.io/hbuilderx.html，下载 App开发版；

1、安装 sass 插件
点击 **工具 => 插件安装 => 安装新插件 => 前往插件市场安装**，在这里你可以搜索自己需要的插件，我们以 sass 为例；找到需要的插件之后点击**下载 => 使用Hbuilderx 导入插件**，这里需要登录 sass 的网站，如果登录成功则会打开 Hbuilderx 编译器，然后点击确定就可以安装了；

3、新建 uni-app项目

Hbuilderx 点击 **文件=>新增=>项目**，本文新建一个小程序项目：uni-app => 填写项目名称、选择项目存放路径 => 模板 uni-ui 项目=>创建，然后就可以生成一个小程序项目；



> components：组件文件
> pages：页面文件
> static：静态文件
> uni_modules：依赖包
> App.vue：应用配置，配置小程序全局样式、生命周期
> main.js：Vue 初始化入口文件
> manifest.json：配置应用名称 appid、logo、版本等打包信息
> pages.json：配置页面路径、页面样式、tabBar等信息
> uni,scss：全局样式

4、把项目运行到微信开发者工具

1、配置 appid

在 **manifest.json 文件 => 微信小程序配置** 填写微信小程序 appID；

2、在 Hbuilderx 配置微信开发者工具的安装路径：这样可以在 Hbuilderx 里面运行的时候自动打开微信开发者工具查看项目
工具 => 设置 => 运行配置 => 小程序运行配置 配置微信开发者工具的安装路径，如：C:\Program Files (x86)\Tencent\微信web开发者工具

3、在微信开发者工具开启服务端口
设置 => 安全设置 => 安全 开启服务端口

4、运行
Hbuilderx 点击 **运行=>运行到小程序模拟器** 点击第一个就可以在 Hbuilderx 自动编译，成功之后会自动打开微信开发者工具；

注意：这个时候我们想修改项目里面的内容，需要在 Hbuilderx 里面修改，例如修改配置：**manifest.json 文件 => 源码视图**

二、实现tabBar效果

1、创建tabBar页面

在 pages 下面创建，右键新建页面，这里创建的是 tanBar 对应的几个页面；记住这里要勾选"创建同名目录、在pages.json 中注册"两个选项，默认是选中的；（home、cate、cart、my）

2、配置tabBar

在 pages.json 文件中在 pages 平级新增 tabBar 的配置：

```
"tabBar":{
    "selectedColor":"#C00000",
    "list":[
        {
            "pagePath":"pages/home/home",
            "text":"首页",
            "iconPath":"static/c1.png",
            "selectedIconPath":"static/c2.png"
        },
        {
            "pagePath":"pages/cate/cate",
            "text":"分类",
            "iconPath":"static/c3.png",
            "selectedIconPath":"static/c4.png"
        },
        {
            "pagePath":"pages/cart/cart",
            "text":"购物车",
```

```
            "iconPath":"static/c5.png",
            "selectedIconPath":"static/c6.png"
        },
        {
            "pagePath":"pages/my/my",
            "text":"我的",
            "iconPath":"static/c7.png",
            "selectedIconPath":"static/c8.png"
        }
    ]
}
```

注意：home 也必须在 pages 数组的第一位；还可以在 pages.json 文件修改 globalStyle 配置项，来自定义自己的导航条样式；

三、配置网络请求

由于小程序不支持 axios ，并且原生的 wx.request() API 功能比较简单，且不支持拦截器等全局定制的功能；所以建议在 uni-app 项目中使用 @escook/request-miniprogram 第三方包发起网络请求；

官网：https://www.npmjs.com/package/@escook/request-miniprogram

1、依照官网提示安装、导入、使用

```
//引入网络请求
import { $http } from '@escook/request-miniprogram';
uni.$http = $http;
//全局baseUrl
$http.baseURL = "https://www.uinav.com";
//请求拦截器
$http.beforeRequest = function (options) {
  uni.showLoading({
    title:"数据加载中..."
  })
}
//相应拦截器
$http.afterRequest = function () {
  uni.hideLoading()
}
```

在 uni-app 中可以使用 uni.xxx 来调用 wx.xxx 的 api；

2、实战

```
//home
data() {
    return {
        swiperList:[]
    };
},
onLoad() {
    this.getSwiperList();
},
methods:{
    async getSwiperList(){
        let { data:res } = await uni.$http.get('接口地址')
        if(res.meta.status !== 200){
            return uni.showToast({
                title:"数据请求失败",
                duration:1500,
                icon:"none"
            })
        }
        this.swiperList = res.message;
    }
}
```

四、uni-app 里面小程序分包

1、创建分包目录

在根目录下创建分包目录，subpackage；

2、在 pages.json 文件中配置

在 pages 节点同级，声明 subpackages 节点用来配置分包结构；

```
"subPackages":[
    {
        "root":"subpackage",
        "pages":[]

```

```
    }
  ]
```

3、创建分包页面

在 sunpackage 目录上右键点击**新建文件**，填写页面名称、选择分包 sunpackage，然后创建就可以了，编译器会自动在代码中将配置信息填充到 sunpackage 分包下面；

```
"subPackages":[
        {
            "root":"subpackage",
            "pages":[{
                    "path" : "goods_detail/goods_detail",
                    "style" :{
                        "navigationBarTitleText": "",
                        "enablePullDownRefresh": false
                    }
                }
            ]
        }
    ]
```

注意：这里提一下，页面跳转传参跟小程序原生跳转传参是一样的：1、navigator 配合 url 跳转、路径拼接传参；2、点击事件通过 uni.redirectTo ；

五、公用方法封装

这里以 错误提示信息为例，在 main.js 中；

```
uni.$showMsg = function(titie="请求失败",duration=1000){
    uni.showToast({
        title,
        duration,
        icon:"none"
    })
}
```

六、搜索功能

1、搜索组件

1、自定义搜索组件：在 components 文件夹右键，选择 **新建组件** ，在这里可以编写组件的内容；

2、小程序自定义组件自定义事件：由于小程序提供的组件已经帮助我们封装了 click 事件，所以我们可以直接使用，但是我们自定义的组件没有封装所以不能直接在自定义的组件上使用 click 事件；

我们可以在父组件绑定一个自定义事件，然后子组件绑定 click 事件，在触发 click 的时候通过 $emit 来触发父组件绑定的自定义事件，这样就可以完成自定义组件的事件传递；

3、吸顶：主要是利用 position:sticky ,把组件定位到页面的顶部；

最后使用组件：直接在页面使用就可以了，组件名是自定义组件的文件名称；

```
//自定义组件
<template>
    <view class="my-search-container" :style="{'background-color':bgColor}">
        <view class="my-search-box" :style="{'border-radius':radius}">
            <uni-icons type="search" size="18"></uni-icons>
            <text class="placeholder">搜索</text>
        </view>
    </view>
</template>
<script>
    export default {
        name:"my-search",
        props:{
            bgColor:{
                type:String,
                default:"#c00000"
            },
            radius:{
                type:String,
                default:"18px"
            }
        },
        methods:{
            //通过 $emit 来触发父组件上绑定的自定义事件
            searchEvent(){
                this.$emit('myclick')
            }
        }
    }
</script>
<style lang="scss">
.my-search-container{
    height: 50px;
    // background-color: #c00000;
    display:flex;

```

```
    align-items: center;
    padding: 0 10px;
    .my-search-box{
        height: 36px;
        background-color: #FFF;
        // border-radius: 18px;
        width: 100%;
        display: flex;
        justify-content: center;
        align-items: center;
        .placeholder{
            font-size: 15px;
            margin-left: 5px;
        }
    }
}
</style>
```

```
//父组件
<template>
    <view>
        <view class="suckTop">
            <my-search @myclick="goSearch" :radius="'0px'" :bgColor="'pink'"
        </view>
    </view>
</template>
<script>
    export default {
        methods:{
            goSearch(){
                uni.navigateTo({
                    url:"/subpackage/search/search"
                })
            }
        }
    }
</script>
<style lang="scss">
.suckTop{
    position: sticky;
    top: 0;
    z-index: 999;

```

```
}
</style>
```

2、搜索建议实现

```
<template>
    <view>
        <view class="suckTop">
            <uni-search-bar @input="input" :radius="18" :focus="true" cancelB
        </view>
    </view>
</template>
<script>
    export default {
        data() {
            return {
                timer:null,
                kw:''
            }
        },
        methods: {
            //输入框事件
            input(e){
                clearTimeout(this.timer)
                this.timer = setTimeout(_=>{
                    this.kw = e.value;
                },500)
            },
        }
    }
</script>
<style lang="scss">
.suckTop{
    position: sticky;
    top: 0;
    z-index: 999;
    .uni-searchbar {
        background-color: #c00000
    }
}
</style>
```

使用 uni-app 提供的组件，添加 focus 让界面自动锁定输入框， input 事件添加节流，然后就可以在节流方法里面调用接口来获取并渲染搜索出来的结果；

3、本地存储

```
//存
uni.setStorageSync('kw',JSON.stringify(this.kw));

//onLoad 声明周期中 取
let list = JSON.parse(uni.getStorageSync('kw') || '');

//删除
uni.setStorageSync('kw',[]);
```

4、过滤器

跟 data 平级声明 filters

```
filters:{
    toFixed(num){
        return Number(num).toFixed(2);
    }
}
```

使用的时候直接在界面上

```
<view>{{num | toFixed}}</view>
```

七、上拉加载、下拉刷新

1、上拉加载

在 pages.json 中找到当前页面在 pages 数组中的路径，在 style 中新增 onReachBottomDistance 设置为 150；

在页面中 methods 平级声明一个 onReachBottom 方法来监听页面上拉事件；

```
data() {
    return {
        isLoading:false
```

```
    };
},
methods:{
    getList(){
        //打开节流阀
        this.isLoading = true;
        //获取数据
        let res = .....
        //关闭节流阀
        this.isLoading = false;
    }
},
//监听上拉事件
onReachBottom() {
    //没有更多数据
    if(this.pagenum*this.pagesize >= this.total) return uni.$showMsg('没有更多
    //限流 防止频繁请求
    if(this.isLoading) return;
    //页面自增加一
    this.pagenum++;
    //获取列表数据的方法
    this.getList();
}
```

2、下拉刷新

在 pages.json 中找到当前页面在 pages 数组中的路径，在 style 中新增 enablePullDownRefresh设置为 true；

在页面中 methods 平级声明一个 onPullDownRefresh方法来监听页面上拉事件；

```
methods:{
    getList(cb){
        //打开节流阀
        this.isLoading = true;
        //调用回调函数
        cb && cb();
        //获取数据
        let res = .....
        //关闭节流阀
        this.isLoading = false;
    }

```

```
},
onPullDownRefresh() {
    this.total = 0;
    this.pagenum = 1;
    this.list = [];
    this.isLoading = false;
    //传入回调函数，停止下拉刷新效果
    this.getList(()=> uni.stopPullDownRefresh());
}
```

八、配置 vuex

1、创建文件

根目录创建文件夹 store，在文件夹下创建文件 store.js ;

2、初始化store

```
//store.js
import Vue from "vue";
import Vuex from "vuex";
Vue.use(Vuex);
import cart from '@/store/cart.js'
const store = new Vuex.Store({
    modules:{
        cart
    }
})
export default store;
```

3、main.js 中引入

```
//main.js
import store from './store/store.js'
const app = new Vue({
    ...App,
    //挂载到vue实例上
    store
})
app.$mount()
```

4、新建模块

新建一个 cart 模块的 js 文件，然后在 store.js 里面引入；

```
//cart.js
export default {
    namespaced:true,
    state:{
        cart:[]
    },
    //修改state 只能通过 mutations
    mutations:{
        addCart:(state,data)=>{
            state.cartList.push(data)
        }
    },
    getter:{}
}
```

5、使用

```
import { mapState, mapActions, mapMutations } from 'vuex';

computed:{
    ...mapState({
        cartList:state=>state.cart.cartList
    })
},
methods: {
    ...mapMutations(['addCart']),
    addCartInfo(){
        this.addCart(2)
        // this.$store.commit('addCart',2)
    },
    //输入框事件
    input(e){
        clearTimeout(this.timer)
        this.timer = setTimeout(_=>{
            this.kw = e.value;
        },500)
    }
    }
```

九、登录

在调用登录接口之前，我们需要先获取用户的基本信息以及 code，作为参数；

1、获取用户基本信息

```
<button open-type="getUserInfo" @getuserinfo="getInfo">一键登录</button>
methods:{
    //自定义方法
    getInfo(e){
        console.log(e)
    }
}
```

这里直接使用 button 组件提供的 open-type 等于 getUserInfo ， 并配合 @getuserinfo 事件绑定的方法中获取到用户信息；这里是固定写法；参考官网：
https://uniapp.dcloud.io/component/button.html

2、获取用户登录凭证 code

这个可以直接调用 uni.login() API ；

```
async getCode(){
    let [err,res] = await uni.login().catch(err=>err)
    console.log(res)
}
```

10、支付

1、请求头添加 token

只有登录成功之后才能调用支付相关的接口，我们需要将登录后获取的 token 设置在有权限的接口请求字段里；一般在请求拦截里面为接口统一配置 header；

```
$http.beforeRequest = function (options) {
  uni.showLoading({
    title:"数据加载中..."
  })
  options.header = {
      Authorization: token
  }
}
```

2、微信支付流程

1、创建订单
将订单信息提交给后台服务器，创建订单，获取订单号；
2、订单预支付
将订单号发送给后台服务器，获取到支付相关的参数；
3、调用微信支付
调用 uni.requestPayment(OBJECT) API，发起支付请求；通过 fail、 success 回调函数监听支付是否成功，然后调取后台接口将支付状态同步给数据库；

十一、发布

小程序只有在发布后才能够被用户检索使用，开发期间为了便于调试，小程序会携带 sourcemap 等类型的文件，并且代码没有进行压缩因此体积比较大，所以需要压缩发布；

1、发布为小程序

1、点击 hbuilderx 工具栏 **发行 => 小程序-微信**，这时候会有一个弹出框，需要填写小程序的名称和 AppID；
2、点击发行按钮，hbuilderx 的控制台就会显示打包编译进度；
3、编译成功之后会自动发开微信开发者工具，这时候点击上传按钮；
4、然后会弹出一些提示信息，点击确定，弹出 版本号、项目备注弹窗，点击上传就可以了；
5、然后微信小程序后台审核上线就可以了；

2、发布为安卓APP

1、点击 hbuilderx 左下角未登录，进行账号登录；
2、登录之后点击项目的 manifest.js 文件，基础配置里面，填写 uni-app 的 AppID 、应用名称、描述、版本等；
3、App图标配置，预览选择图片，然后自动生成配置；
4、点击 hbuilderx 工具栏 **发行 => 原生App-云打包**，然后弹出弹窗，选择 安卓apk包，ios需要先购买开发者身份才能打包，使用公共的测试证书，勾选打正式包，然后点击打包；
5、然后在 hbuilderx 控制台显示打包进度，成功之后会返回一个**下载地址**的连接，点击连接就可以下载 apk 的安装包，这个 apk 包就可以在安卓系统上安装查看了；

其他

下面是一些比较常见的组件、API、问题：
1、API：uni.previewImage(OBJECT)
预览图片，可以将轮播图方法查看；
2、API：uni.chooseAddress(OBJECT)
获取用户收货地址。调起用户编辑收货地址原生界面，并在编辑完成后返回用户选择的地址，需要用户授权 scope.address；
3、组件：rich-text

渲染富文本；
4、组件：uni-goods-nav
商品加入购物车，立即购买组件；
5、问题：.webp 后缀图片在 ios 不展示问题
ios 上图片 .webp 格式支持不怎么友好，可以只要正则表达式将图片后缀名替换成 .jpg ；
6、问题：商品价格闪烁问题
数据在请求到页面之前， data 里面的数据初始为 {} ，因此初次渲染会导致一些数据闪烁，可以先利用 v-if 判断这个数据是否存在，来控制整体界面的显示隐藏；
7、问题：收获地址授权失败问题
判断是否是授权失败问题，是则直接调用 uni.openSetting(OBJECT) API 开启地址权限；注意兼容 ios 和 安卓；