

概述

本文档旨在为集团领域、分子公司、ISV 等现存的基于 Summer、Cordova、H Builder 等技术打包的独立 APP，向 YonBuilder 移动平台迁移提供整体技术方案，促进 YonBIP 移动开发技术的统一、平台与领域产品的分层研发与设计。

第一章 通用迁移标准

现存独立 APP 向 YonBuilder 迁移,核心涉及 HTML5 代码和迁原生 API 两个关键部分的迁移。

其中所有基于标准 HTML5 开发的 JS、CSS、HTML 代码可无缝迁移；使用到原生 API 的功能，则需要替换为 MTL.js 中对应的同等功能接口，如果 MTL.js 中未提供相关接口，则调用 API.js 中的接口，如果 MTL.js 和 API.js 中均未提供的功能，则需通过原生插件扩展实现。

原生 API 调用优先级为：MTL.js > API.js > 原生扩展。

1.1 统一 API 介绍

1.1.1 MTL.js 参考文档

接口文档地址：<https://mtldev.yuque.com/books/share/a729fe5d-b5e7-4e7e-a1b9-3d428d750ed2/gwxhbu>

1.1.2 MTL.js 引入方式

(1) MTL npm 引入，适用于基于 react 方式开发的离线应用。

引入方式：

```
npm install mtl-js-sdk
```

在需要调用 api 的页面 import mtl from 'mtl-js-sdk';

(2) MTL cdn 版本，适用于 H5 轻应用。

引入方式：

//引入指定版本号

```
<script src="https://design.yonyoucloud.com/library/mtl/[版本号]/mtl.min.js"></script>
```

//始终引入最新 snapshot 版本-开发版本-一般更新周期：1-3 天

```
<script src="https://design.yonyoucloud.com/library/mtl/snapshot/mtl.min.js"></script>
```

//始终引入最新 beta 版本-体验版本-一般更新周期：1-2 周

```
<script src="https://design.yonyoucloud.com/library/mtl/beta/mtl.min.js"></script>
```

//始终引入最新 release 版本-稳定版本-一般更新周期：1-3 月

```
<script src="https://design.yonyoucloud.com/library/mtl/release/mtl.min.js"></script>
```

```
<script src="https://design.yonyoucloud.com/library/mtl/release/mtl.min.js"></script>
```

1.1.3 API.js 参考文档

接口文档地址：<https://docs.apicloud.com/Client-API/api>

1.1.4 原生插件扩展参考文档

文档地址：<https://yonbip-daily-oss.yyuap.com/yonbuilder-mobile/resource/yybuilding/plugin/dde287cca7574e158ef401a8b783608a/%E6%8F%92%E4%BB%B6%E5%BC%80%E5%8F%91%E6%96%87%E6%A1%A3.zip>

1.2 HTML5 代码迁移

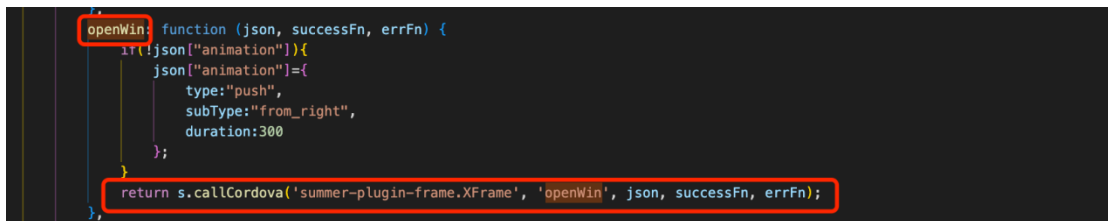
所有基于标准 HTML5 开发的 JS、CSS、HTML 代码都可以直接拷贝迁移，包括基于 react、vue、angular 等前端框架开发的代码。

1.3 原生 API 迁移

现有应用迁移至 YonBuilder 移动打包平台，另一个重要的工作是其他平台本身扩展的原生 API 或者插件的迁移和适配。

1.3.1 summer.js 迁移

方式一：修库，即修改 summer.js 库，summer.js 中有大量原生 api，需要修改成 mtl 或者 api，如下图所示，summer.js 的 openWin 调用的是 cordova 接口，需要修改为 api.openWin 接口



```
openWin: function (json, successFn, errFn) {
  if(!json["animation"]){
    json["animation"]={
      type:"push",
      subType:"from_right",
      duration:300
    };
  }
  return s.callCordova('summer-plugin-frame.XFrame', 'openWin', json, successFn, errFn);
},
```

方式二：删库，即直接删除 summer.js 文件的引入，凡是项目有 summer.方式调用的，都需要修改为 mtl 或者 api 调用方式，如下图所示，删除 summer.js 文件的引入，将 summer.openWin 修改为 api.openWin 接口

```
<html>

<head>
  <meta name="viewport" content="width=device-width, minimum-scale=1, maximum-scale=1">
  <meta charset="UTF-8">
  <title></title>
  <link rel="stylesheet" href="css/iuap/font-icons.css">
  <link rel="stylesheet" href="css/iuap/iuapmobile.um.css">
  <link rel="stylesheet" href="css/index.css">
  <script src="env.config.js"></script>
  <script src="js/iuap/summer.js"></script>
  <script src="js/iuap/jquery.min.js"></script>
  <script src="js/iuap/iuapmobile.frameworks.ui.js"></script>
  <script src="js/iuap/font.js"></script>
  <script src="js/cpu/common.js"></script>
  <script src="js/index.js"></script>
  <script>
    function openTest() {
      openProtoWindow('test', 'html/approve/fe/price analysis1.html', {}, false, '审批详情', null, {});
    }
  </script>
  <style>
    .icon_box {
      margin-bottom: 0.1rem
    }

    .icon_box img {
      width: 0.42rem;
      height: 0.38rem;
    }
  </style>
</head>

}

var reLogin= function() {
  summer.rmStorage("userinfo");
  var params = {
    "method": "YYIM.logout",
    "params": {}
  };
  cordova.exec(null, null, "XService", "callSync", [params]);
  setTimeout(function(){summer.openWin({
    "id": 'login',
    "url": 'html/login/login.html',
    "isKeep": false
  })}, 3000);
}
```

删除

1.3.2 mtl.js 迁移

项目中所引用的 MTL.js 为 2022.3.30 之前的版本，都需要更新到最新版本，最新版本的下载地址，见章节 一 通用迁移标准 中 mtl 的引入

1.3.3 mui.js 迁移

mui.js 迁移即 HBuilder 项目的迁移，项目中凡是有 mui. 或者 plus. 前缀的 API，都需要替换为 MTL.js 或者 API.js 下对应的接口。例如下图中所示的 plus.webview.create 接口是用于打开新窗口，需要替换为 api.openWin。

```
reset
// var wa = plus.nativeUI.showWaiting();
// _openw=plus.webview.create(id, {scrollIndicator: 'none', scalable: false, popGesture: 'hide'}, {preate: true});
preate(id);
// _openw,
// _openw.addEventListener('loaded', function() { // 页面加载完成后才显示
// setTimeout(function() { // 延后显示可避免低端机上动画时白屏
// wa.close();
// _openw.showed=true;
// s|_openw.show(a, null, function() {
// _openw=null; // 避免快速点击打开多个页面
// });
```

1.3.4 cordova.js 迁移

直接删除项目中 cordova 相关的文件，cordova 中封装了原生的 api 接口，调试过程中报错的，直接替换为 mtl 或者 api 下的接口。

```
html • JS cordova.js ×
u > android > app > src > main > assets > www > JS cordova.js > <function> > define("cordova/argscheck") callback
var APP_PLUGIN_NAME = Number(require('cordova').platformVersion.split('.')[0]) >= 4 ? 'CoreAndroid' : 'App';
module.exports = {
  /**
   * Clear the resource cache.
   */
  clearCache: function() {
    exec(null, null, APP_PLUGIN_NAME, "clearCache", []);
  },
  /**
   * Load the url into the webview or into new browser instance.
   *
   * @param url      The URL to load
   * @param props    Properties that can be passed in to the activity:
   *   wait: int      => wait msec before loading URL
   *   loadingDialog: "Title,Message" => display a native loading dialog
   *   loadUrlTimeoutValue: int => time in msec to wait before triggering a timeout error
   *   clearHistory: boolean => clear webview history (default=false)
   *   openExternal: boolean => open in a new browser (default=false)
   *
   * Example:
   * navigator.app.loadUrl("http://server/myapp/index.html", {wait:2000, loadingDialog:"Wait,Loading App", loadUrlTimeoutValue: 60000});
   */
  loadUrl: function(url, props) {
    exec(null, null, APP_PLUGIN_NAME, "loadUrl", [url, props]);
  },
  /**
   * Cancel loadUrl that is waiting to be loaded.
   */
  cancelLoadUrl: function() {
    exec(null, null, APP_PLUGIN_NAME, "cancelLoadUrl", []);
  },
  /**
   * Clear web history in this web view.
   * Instead of BACK button loading the previous web page, it will exit the app.
   */
  clearHistory: function() {
    exec(null, null, APP_PLUGIN_NAME, "clearHistory", []);
  },
};
```

1.3.5 yybrige.js 迁移

应用中凡是通过 YYEsnBridge.do 方式调用的，都需要替换为 mtl 或者 api 下相应的接口，如果 mtl 或者 api 下没有相关的接口，可以通过封装模块来实现。

```
// 只有一个按钮的弹窗
YYEsnBridge.ready(function (){
    YYEsnBridge.do('showAlert', {
        'type': 'alert',
        'title': '温馨提示',
        'content': '消息已发出',
        'confirmButtonText': '我知道了',
        'success': function(res){alert('调用成功');},
        'fail': function(err){alert('调用失败');},
        'complete': function(res){alert('调用完成后执行');}
    });
});
```

1.3.6 插件迁移

项目中如果有第三方插件的需求，不管之前的项目是在 cordova 上还是 hbuilder 上，都需要替换为 mtl 平台上的插件，如果 mtl 平台上没有找到相关的插件，团队中如果原生开发人员可以根据文档（<https://yonbip-daily-oss.yyuap.com/yonbuilder-mobile/resource/yybuilding/plugin/dde287cca7574e158ef401a8b783608a/%E6%8F%92%E4%BB%B6%E5%BC%80%E5%8F%91%E6%96%87%E6%A1%A3.zip>）自行封装，如果没有可以协调 iuap 移动开发部的同事进行开发。

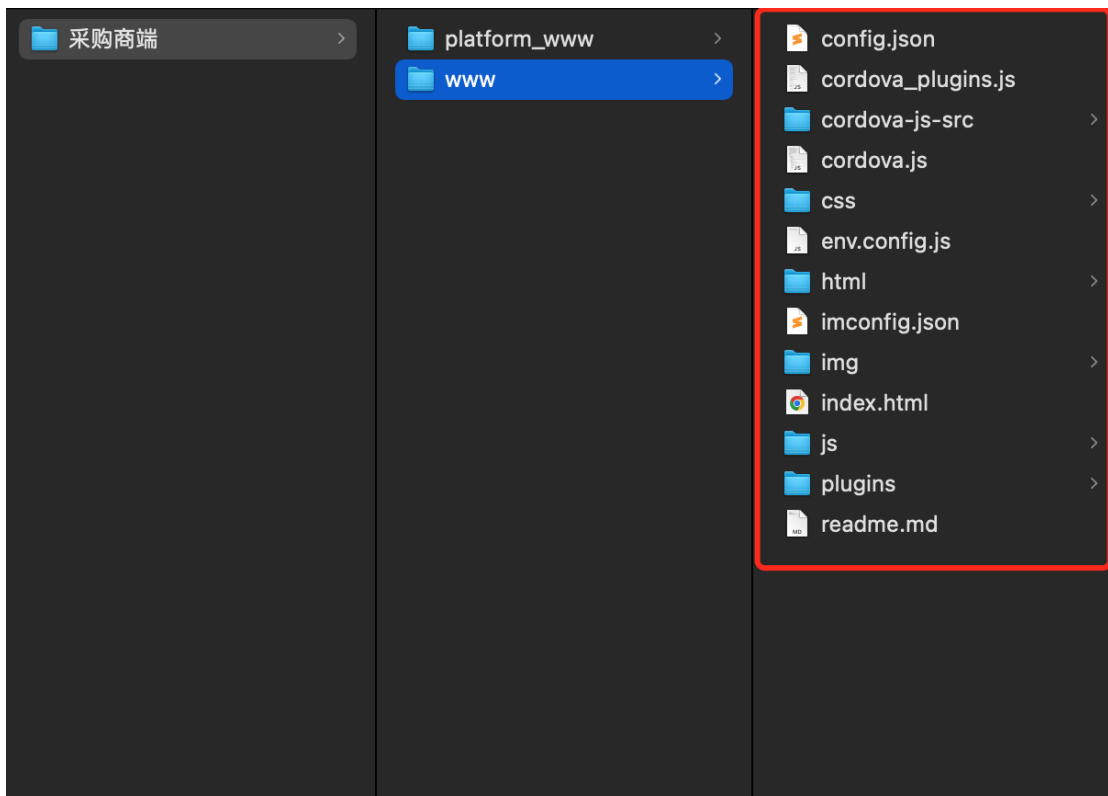
第二章 Summer 应用迁移

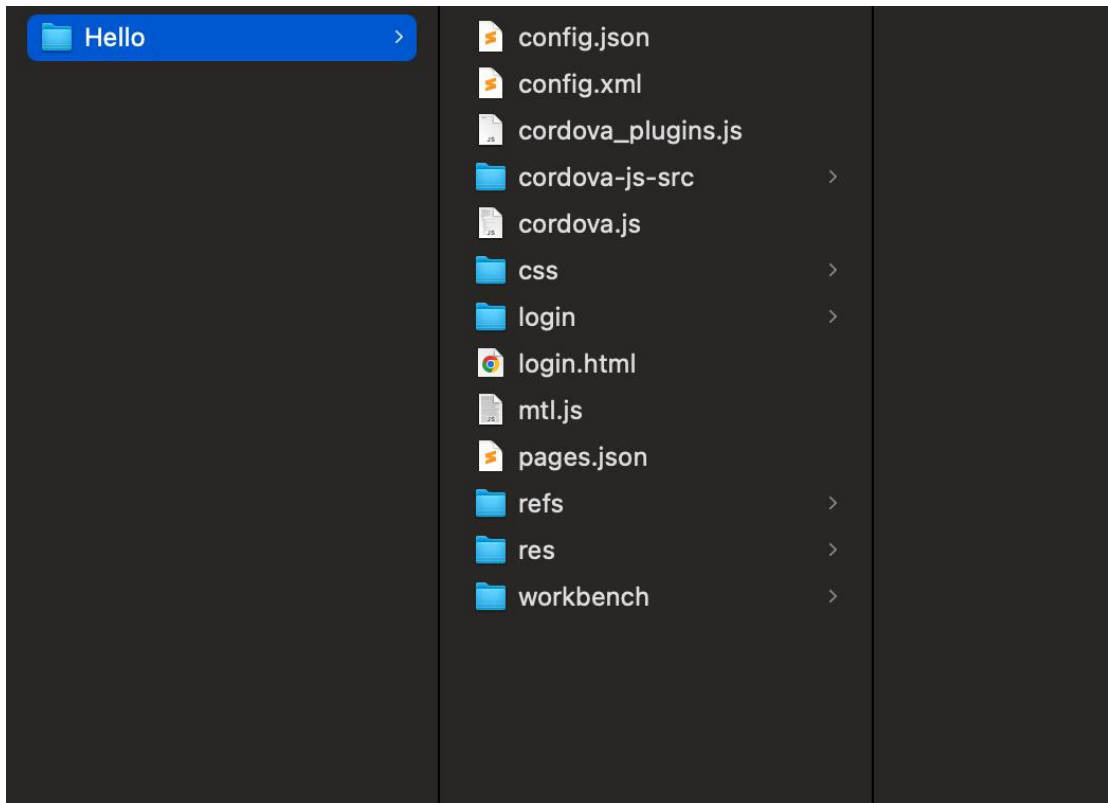
2.1 迁移步骤

(1) 在 APICloud Studio 3 新建一个空白应用

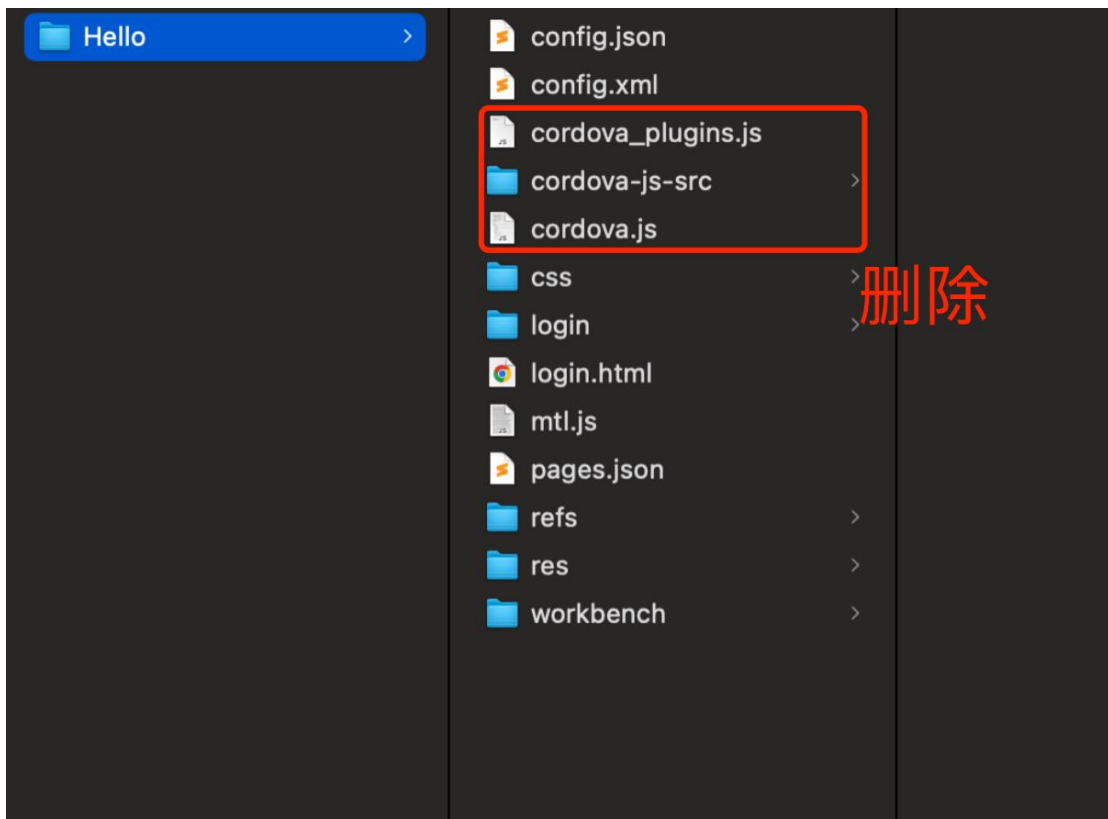


(2) 将原项目下根目录/www/下的文件拷贝到第一步创建的项目根目录中





(3) 删除项目中 cordova 相关的文件



(4) 打开 config.xml 文件，将启动页改为项目中的启动页

```
<widget id="A6116968898077" version="0.0.1">
  <name>Hello</name>
  <description>
    Example For APICloud.
  </description>
  <author email="developer@apicloud.com" href="http://www.apicloud.com">
    Developer
  </author>
  <content src="index.html" />
  <access origin="*" />
</widget>
```

2.2 调试方式

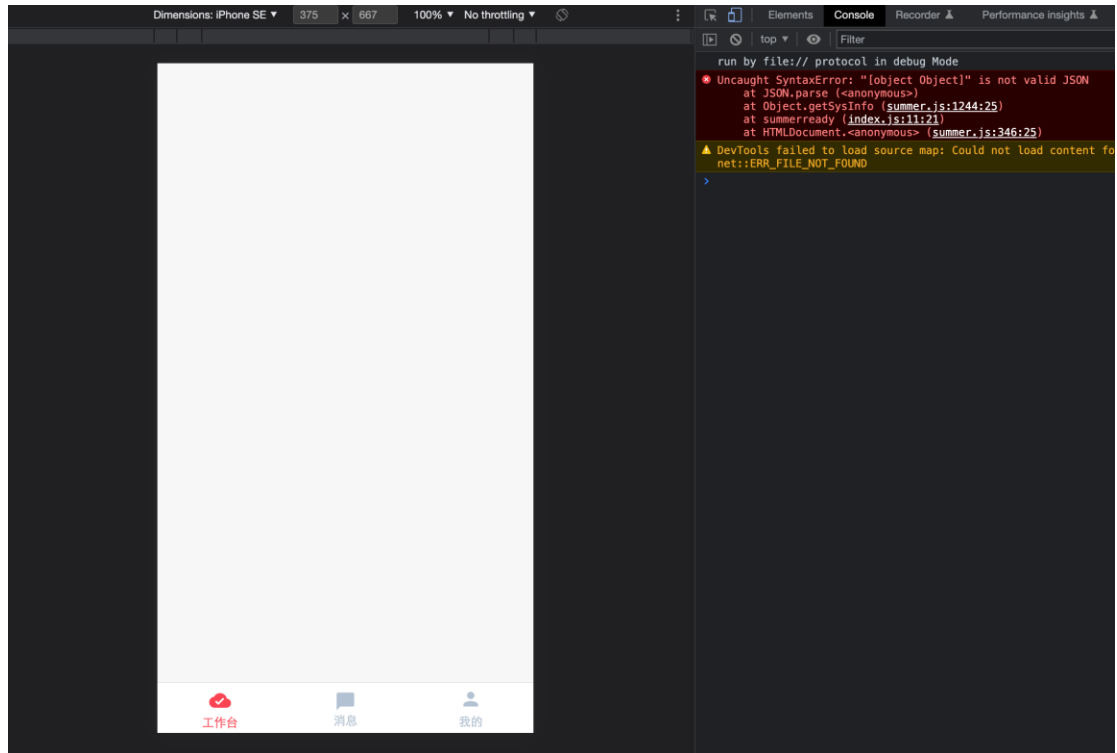
(1) 登录 apicloud 控制台，打开对应项目-模块-自定义 loader-编译，将 loader 安装到手机上。

(2) 利用 APICloud Studio 3 的 WI-FI 调试和 USB 调试功能，将项目在真机上调试，项目运行过程中如遇到问题，就需要更换接口，首先引入 MTL.js，查看 mtl 中有没有相关的接口(接口文档：<https://mtldev.yuque.com/books/share/a729fe5d-b5e7-4e7e-a1b9-3d428d750ed2/gwxhbu>)，如果 MTL.js 中没有相应接口，可以调用 API.js 中的接口(接口文档：<https://docs.apicloud.com/Client-API/api>)，如果有原生功能，可以封装模块，具体的封装文档参考：<https://yonbip-daily-oss.yyuap.com/yonbuilder-mobile/resource/yybuilding/plugin/dde287cca7574e158ef401a8b783608a/%E6%8F%92%E4%BB%B6%E5%BC%80%E5%8F%91%E6%96%87%E6%A1%A3.zip>。

(3) 项目启动后，APICloud Studio 3 的控制台会输出大量 cordova 接口报错的相关信息，找到对应位置，将相关接口替换为 mtl 或者 api 接口。

2.3 chrome 调试方式

SPA 应用可以通过 chrome 开发者调试工具进行调试，如果依赖了原生 api，可以通过真机安装 apploader 或自定义 loader 进行调试。



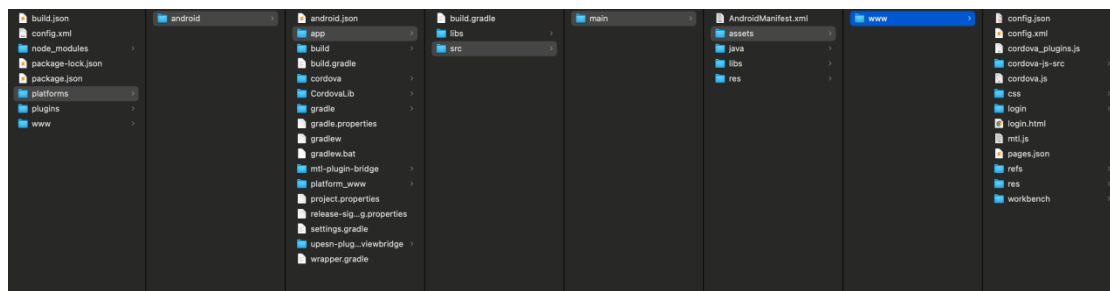
第三章 Cordova 应用迁移

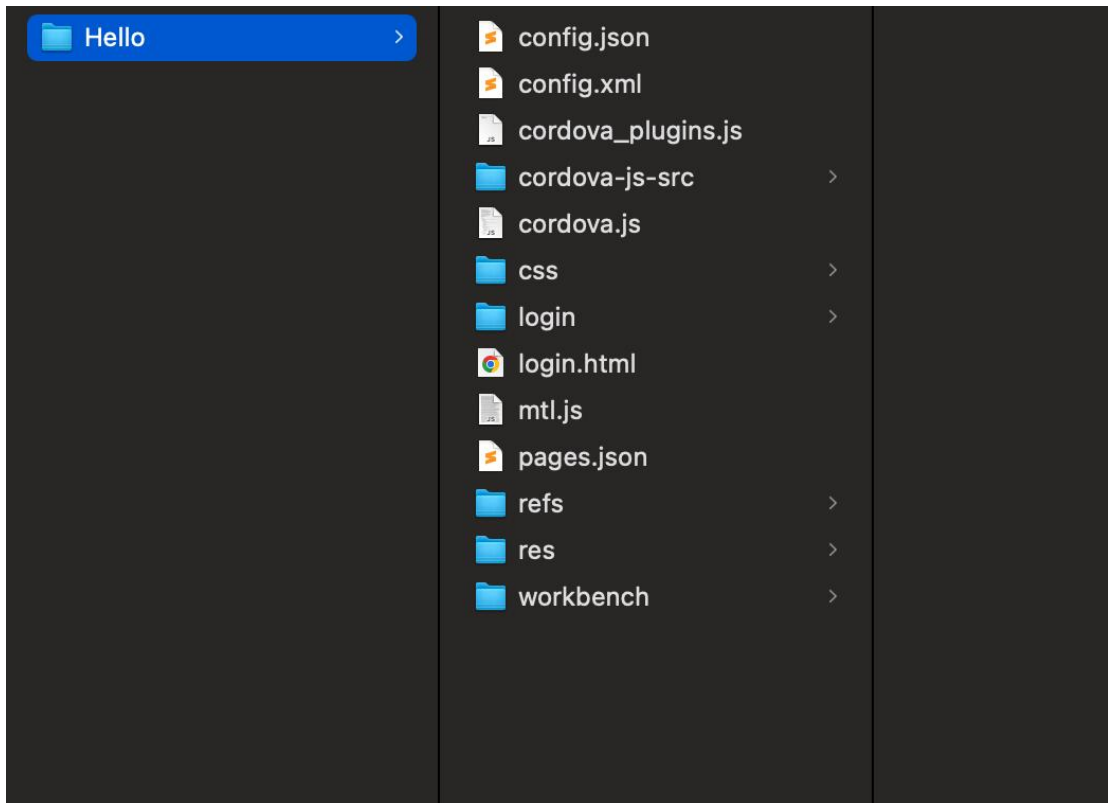
3.1 迁移步骤

(1) 在 APICloud Studio 3 新建一个空白应用。

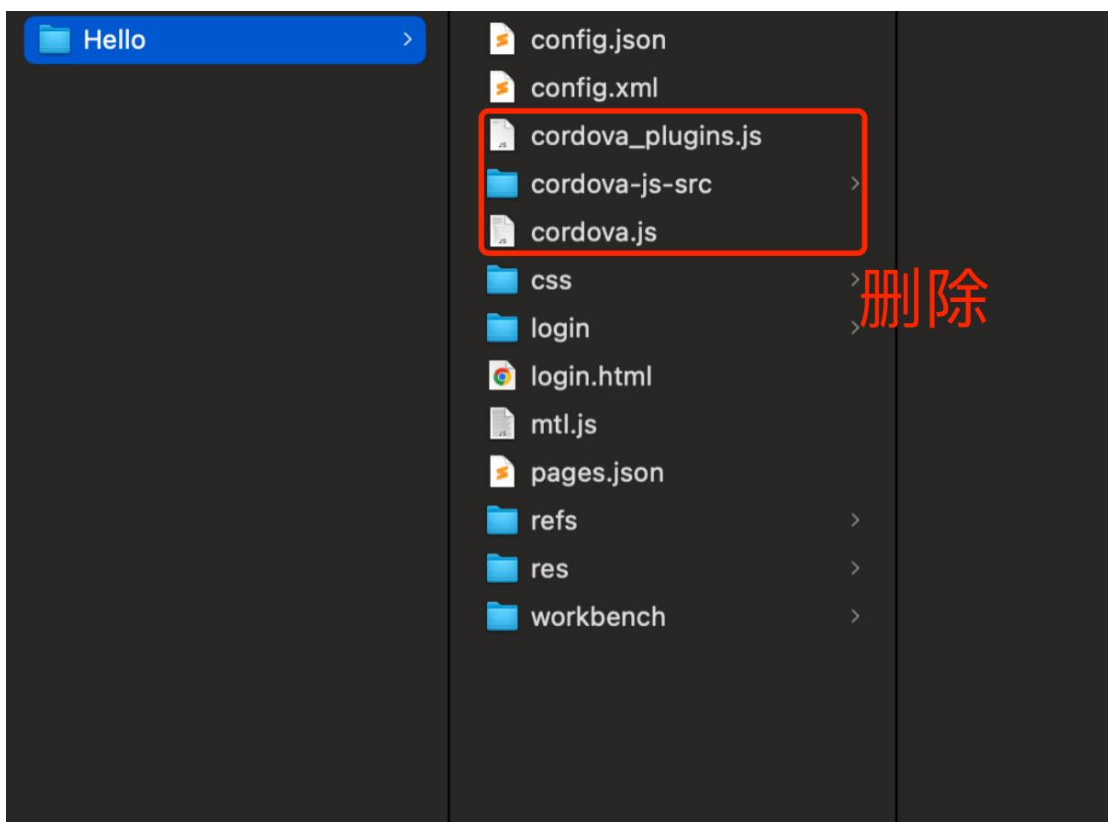


(2) 将原项目下 assets/www/下的文件拷贝到第一步创建的项目根目录中。





(3) 删除项目中 cordova 相关的文件。



(4) 打开 config.xml 文件，将启动页改为项目中的启动页。

```
<widget id="A6116968898077" version="0.0.1">
  <name>Hello</name>
  <description>
    Example For APICloud.
  </description>
  <author email="developer@apicloud.com" href="http://www.apicloud.com">
    Developer
  </author>
  <content src="login.html" />
  <access origin="*" />
  <preference name="pageBounce" value="false"/>
  <preference name="appBackground" value="rgba(0,0,0,0.0)"/>
  <preference name="windowBackground" value="rgba(0,0,0,0.0)"/>
  <preference name="frameBackgroundColor" value="rgba(0,0,0,0.0)"/>
  <preference name="hScrollBarEnabled" value="false"/>
</widget>
```

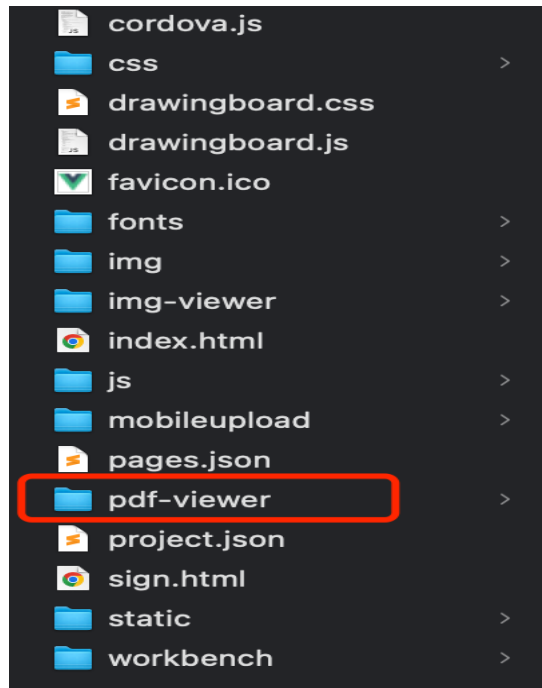
3.2 调试方式

(1) 登录 apicloud 控制台，打开对应项目-模块-自定义 loader-编译，将 loader 安装到手机上。

(2) 利用 APICloud Studio 3 的 WI-FI 调试和 USB 调试功能，将项目在真机上调试，项目运行过程中如遇到问题，就需要更换接口，首先引入 MTL.js，查看 mtl 中有没有相关的接口(接口文档：<https://mtldev.yuque.com/books/share/a729fe5d-b5e7-4e7e-a1b9-3d428d750ed2/gwxhbu>)，如果 MTL.js 中没有相应接口，可以调用 API.js 中的接口(接口文档：<https://docs.apicloud.com/Client-API/api>)，如果有原生功能，可以封装模块，具体的封装文档参考：<https://yonbip-daily-oss.yyuap.com/yonbuilder-mobile/resource/yybuilding/plugin/dde287cca7574e158ef401a8b783608a/%E6%8F%92%E4%BB%B6%E5%BC%80%E5%8F%91%E6%96%87%E6%A1%A3.zip>。

3.2.1 调试举例

1. 在 cordova 框架开发应用中，原生中可能封装了某些功能，供前端调用，迁移到 mtl 中，可以用 mtl 提供的模块来实现这些功能。例如，项目中有查看 PDF 文件的功能，则可以引用 mtl 模块库中 PDF 相关的模块来完成功能。



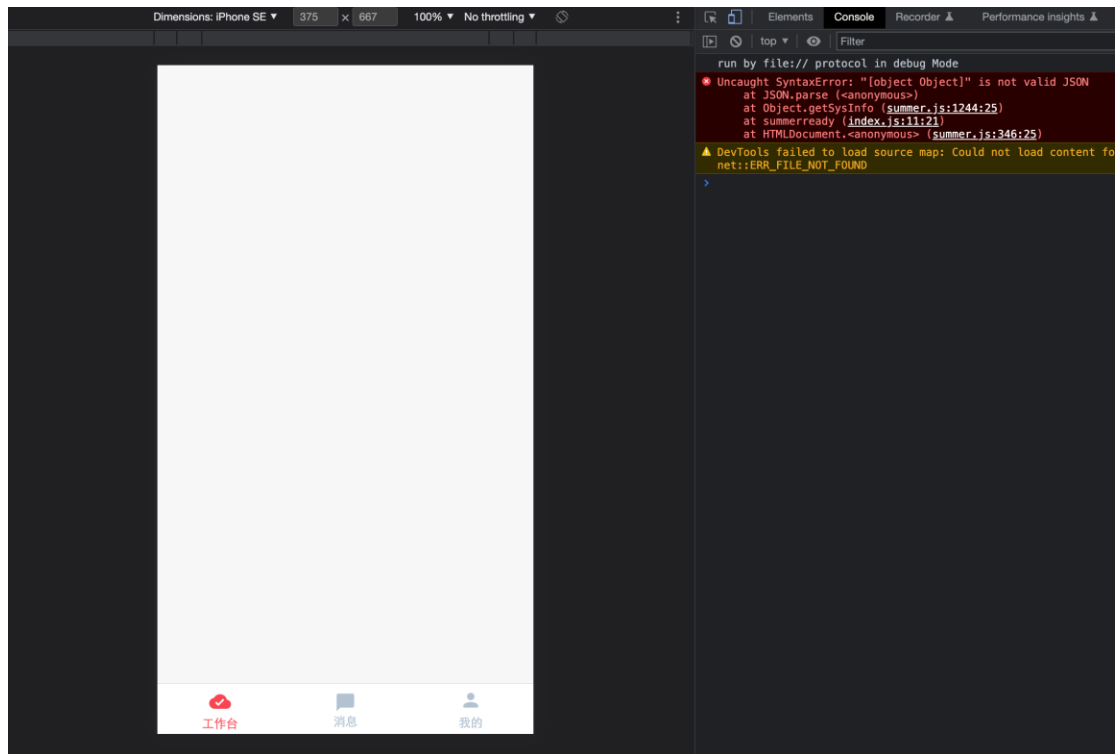
系统 自定义

全部 设备访问 功能扩展 开放SDK 人工智能 直播/音视频 轻应用

序号	插件名称	描述	系统	最新版本	分类	类型	更新时间	插件文档
1	pdf阅读器	手机怎么打开pdf文件_PDF阅读器	Android、IOS	1.2.2	功能扩展	原生	2022-03-30	帮助文档
2	安卓pdf阅读器	androidPdfReader 是一个简单的pdf阅读器	Android	1.0.0	功能扩展	原生	2022-03-30	帮助文档

3.3 chrome 调试方式

SPA 应用可以通过 chrome 开发者调试工具进行调试，如果依赖了原生 api，可以通过真机安装 apploader 或自定义 loader 进行调试。



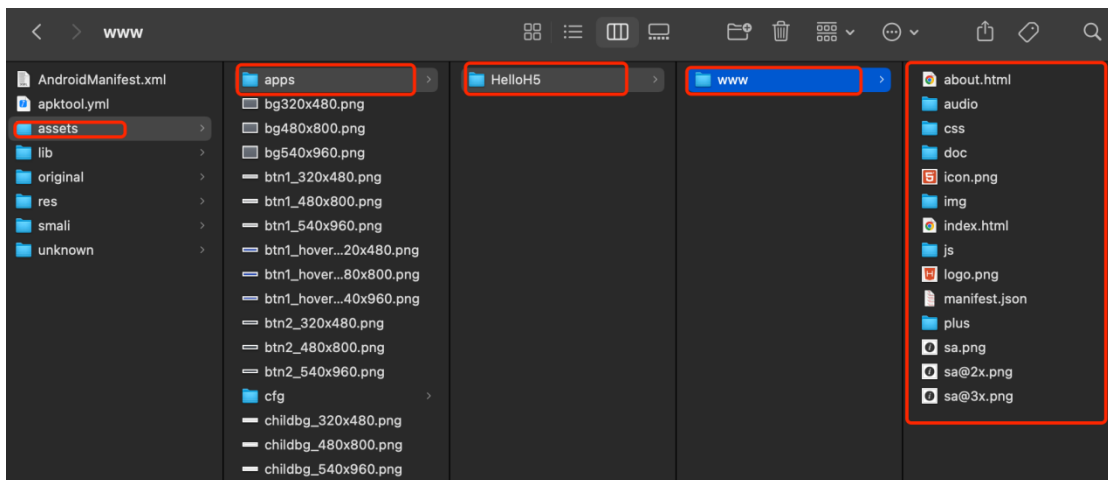
第四章 HBuilder 应用迁移

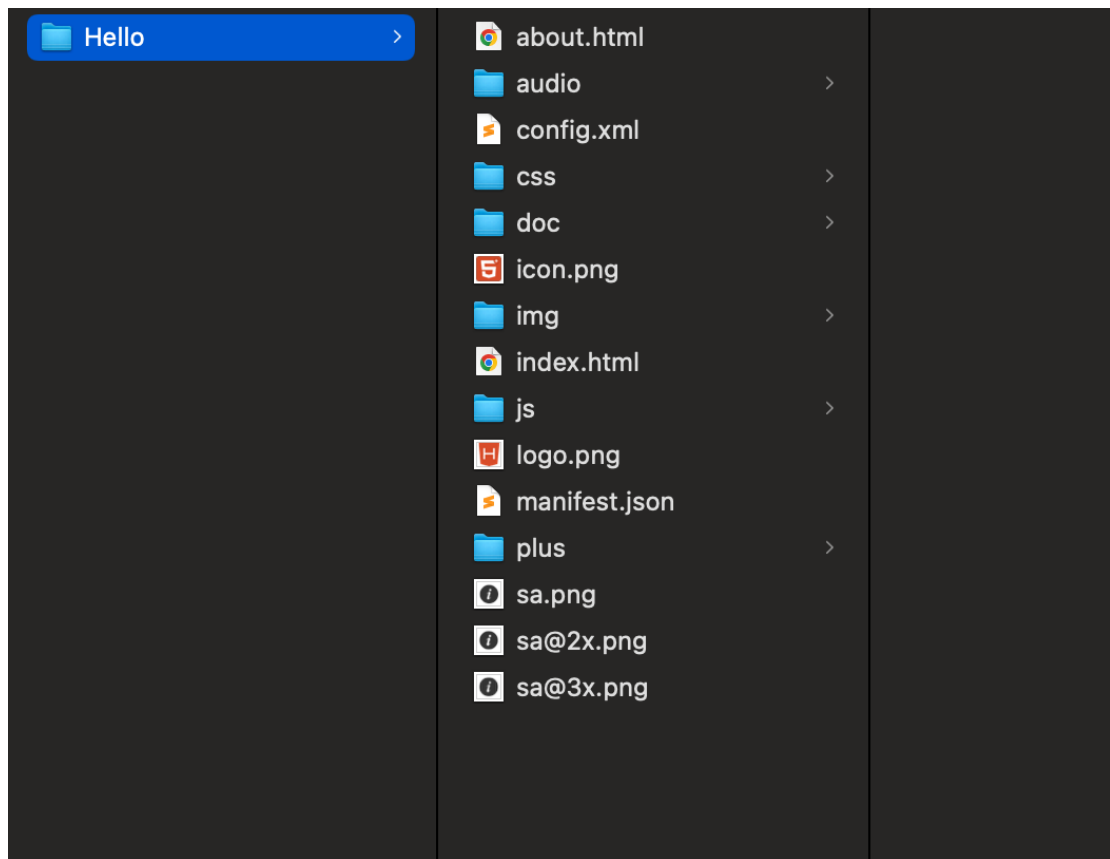
4.1 迁移步骤

(1) 在 APICloud Studio 3 新建一个空白应用。



(2) 将原项目下 assets/apps/项目名/www/下的文件拷贝到第一步创建的项目根目录中。





(3) 打开 config.xml 文件，将启动页改为项目中的启动页。

```
<widget id="A6116968898077" version="0.0.1">
  <name>Hello</name>
  <description>
    Example For APICloud.
  </description>
  <author email="developer@apicloud.com" href="http://www.apicloud.com">
    Developer
  </author>
  <content src="index.html" />
  <access origin="*" />
  <preference name="pageBounce" value="false"/>
  <preference name="appBackground" value="rgba(0,0,0,0.0)"/>
  <preference name="windowBackground" value="rgba(0,0,0,0.0)"/>
  <preference name="frameBackgroundColor" value="rgba(0,0,0,0.0)"/>
  <preference name="hScrollBarEnabled" value="false"/>
  <preference name="vScrollBarEnabled" value="false"/>
  <preference name="autoLaunch" value="true"/>
  <preference name="fullScreen" value="false"/>
  <preference name="autoUpdate" value="true" />
  <preference name="smartUpdate" value="false" />
  <preference name="debug" value="true"/>
  <preference name="statusBarAppearance" value="true"/>
</widget>
```

4.2 调试方式

(1) 登录 apicloud 控制台，打开对应项目-模块-自定义 loader-编译，将 loader 安装到手机上。

(2) 利用 APICloud Studio 3 的 WI-FI 调试和 USB 调试功能，将项目在真机上调试，项目运行过程中如遇到问题，就需要更换接口，首先引入 MTL.js，查看 mtl 中有没有相关的接口（接口文档：<https://mtldev.yuque.com/books/share/a729fe5d-b5e7-4e7e-a1b9-3d428d750ed2/gwxhbu>），如果 MTL.js 中没有相应接口，可以调用 API.js 中的接口（接口文档：<https://docs.apicloud.com/Client-API/api>），如果有原生功能，可以封装模块，具体的封装文档参考：<https://yonbip-daily-oss.yyuap.com/yonbuilder-mobile/resource/yybuilding/plugin/dde287cca7574e158ef401a8b783608a/%E6%8F%92%E4%BB%B6%E5%BC%80%E5%8F%91%E6%96%87%E6%A1%A3.zip>。

4.2.1 调试举例

1. 项目中所有用到 mui. 或者 plus. 前缀的 API 调用，都需要替换为 mtl 下对应的接口。例如：项目中用到了 plus.nativeUI.alert 或 mui.alert，需要替换为 mtl.showAlert 接口。

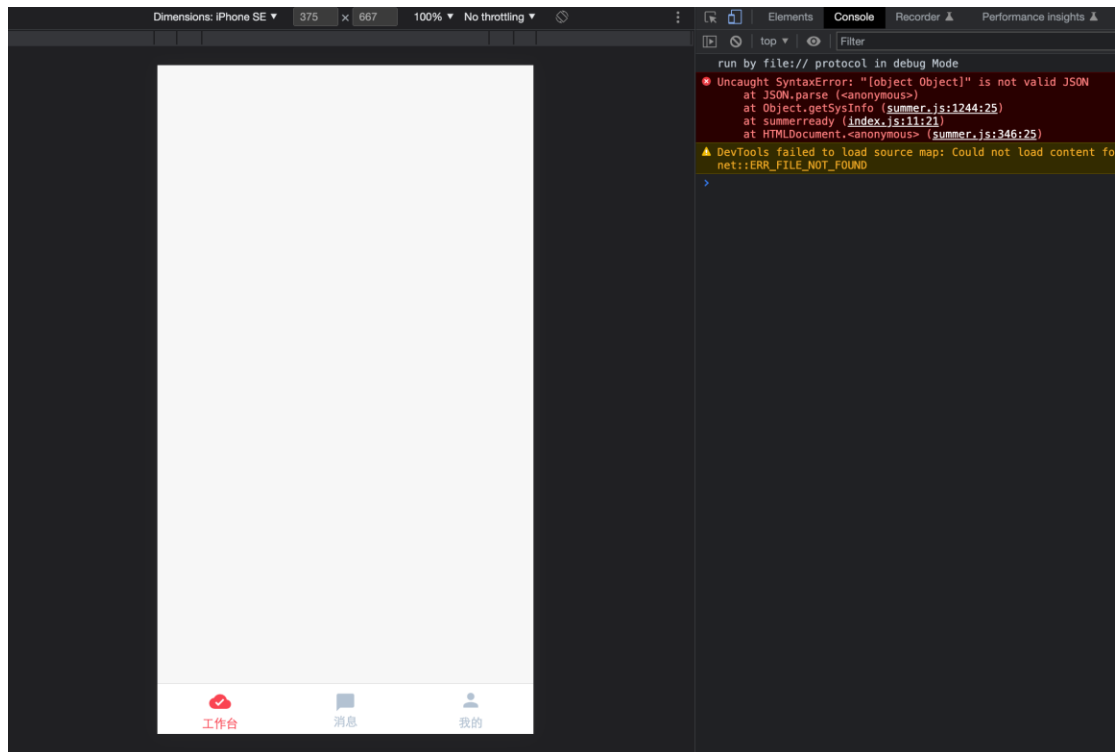
```
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="initial-scale=1.0, maximum-scale=1.0, user-scalable=no" />
  <meta name="HandheldFriendly" content="true" />
  <meta name="MobileOptimized" content="320" />
  <title>Hello H5+</title>
  <script type="text/javascript" src="../js/common.js"></script>
  <script type="text/javascript">
    <script type="text/javascript" src="../js/mtl.js"/>
  </script>
  var img = null;
  var blist = [];
  > function scanned( t, r, f ) {
  function selected( id ) {
    var h = blist[id];
    update( h.type, h.result, h.file );
    if ( h.result.indexOf("http://")==0 || h.result.indexOf("https://")==0 ) {
      plus.nativeUI.confirm( h.result, function ( i ) {
        if ( i.index == 0 ) {
          plus.runtime.openURL( h.result );
        }
      }, "", ["打开", "取消"] );
    } else {
      // plus.nativeUI.alert( h.result );
      mtl.showAlert({
        type: 'alert',
        content:h.result,
        success: function(res) {
        },
        fail: function(err) {
        }
      });
    }
  }
}
```

2. 项目中有用到第三方的能力，比如微信支付或支付宝支付，可以用 mtl 提供的插件，如果 mtl 没有提供相关插件，则可以根据插件开发文档封装。

序号	插件名称	描述	系统	最新版本	分类	类型	更新时间	插件文档
1	用友推送	离线推送, 主流厂商推送	Android, iOS	1.0.0	开放SDK	原生	2022-04-01	帮助文档
2	微信登录、分享	封装了微信开放平台的分享登陆功能	Android, iOS	1.1.3	开放SDK	原生	2022-03-30	帮助文档
3	微信支付	封装了微信开放SDK的支付功能	Android, iOS	1.0.9	开放SDK	原生	2022-03-30	帮助文档
4	微信刷脸支付	微信刷脸支付 (beta版)	Android	1.0.1	开放SDK	原生	2022-03-30	帮助文档
5	新浪微博	微博分享接口,新浪微博授权登陆及分享内容到微博	Android, iOS	1.0.7	开放SDK	原生	2022-03-30	帮助文档

4.3 chrome 调试方式

SPA 应用可以通过 chrome 开发者调试工具进行调试，如果依赖了原生 api，可以通过真机安装 apploader 或自定义 loader 进行调试。



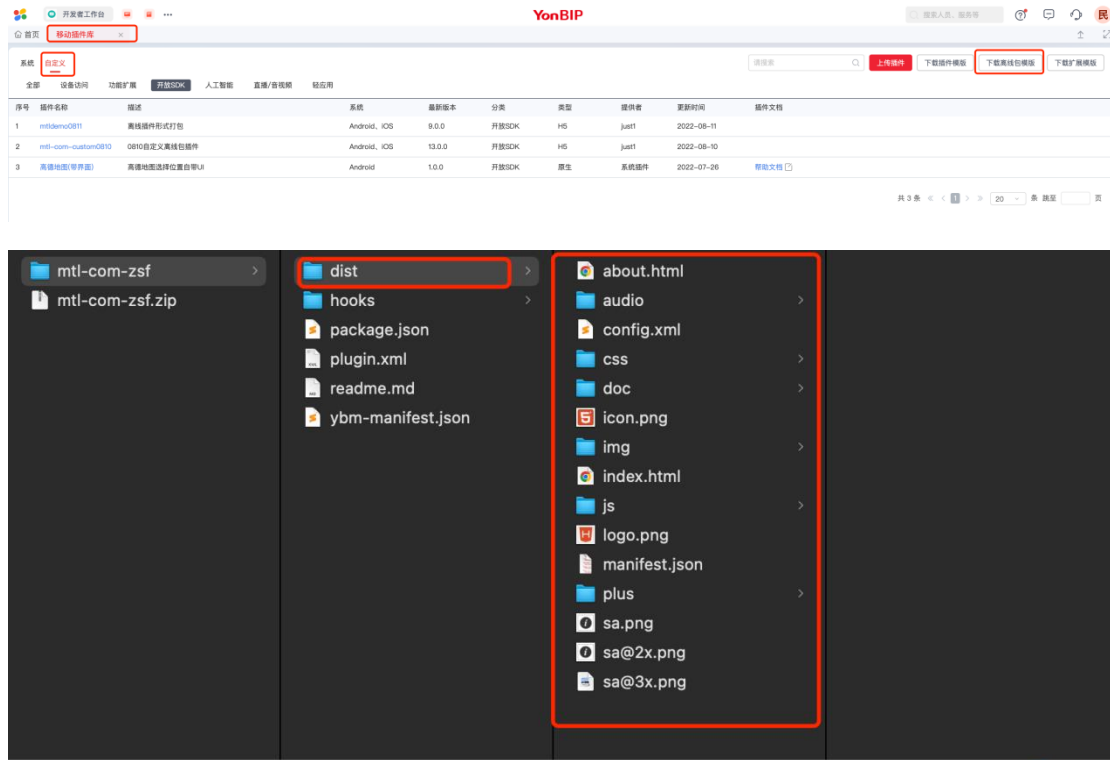
第五章 编译打包

经过上述步骤调试没有问题后就可以在移动工作台打包了。

5.1 制作插件包

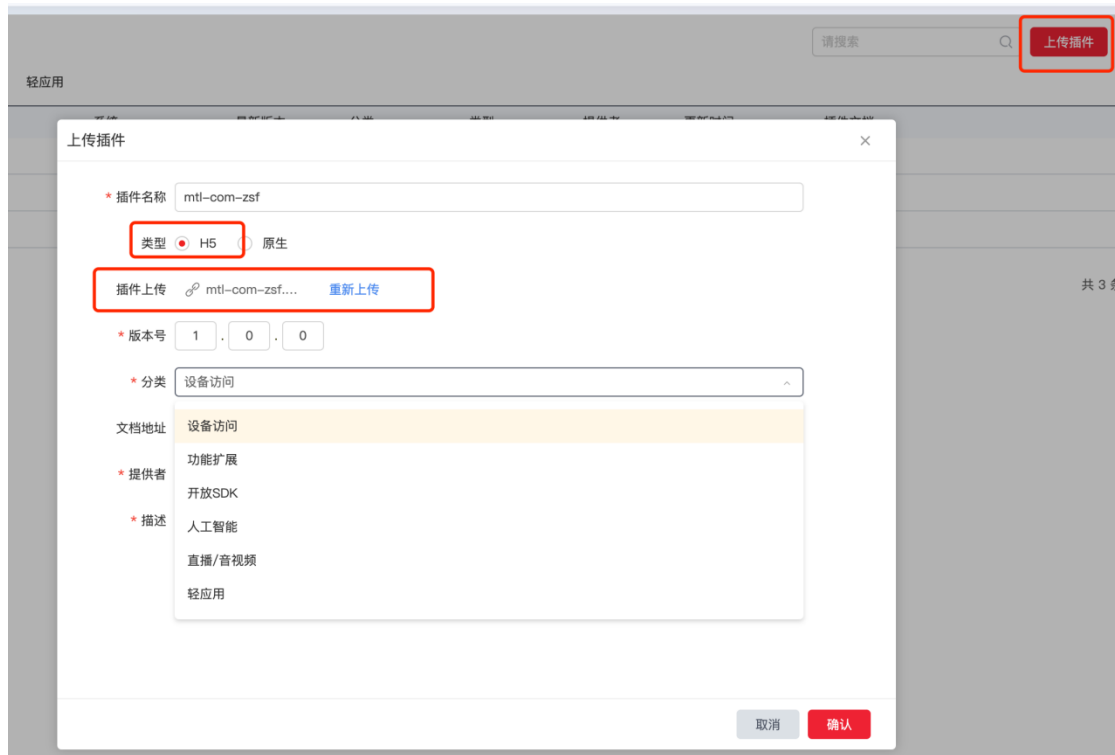
导航到：移动工作台-应用构建服务-移动插件库，选择自定义，下载离线包模版，解压，将 APICloud Studio 3 创建的项目根目录下的文件拷贝到离线包中的 dist 目录下（拷贝之前先删除 dist 目录下的所有文件），文件拷贝完成后，再把离线包压缩成 zip。具体的离线包中的各种参数说明参考：<https://eku1k8-illfov1.yuque.com/docs/share/446d5a27-768e-4d3c-856d-8584ac5564d0?#>。

注：目前 iOS 端打包时使用的离线包插件的名称、id、code 等必须叫 upesn-plugin-workbench 才能生效。



5.2 上传插件包

在移动插件库界面，点击上传插件，输入插件名称，类型选择 h5，上传 5.1 制作的插件，其他的配置按照要求填写即可。

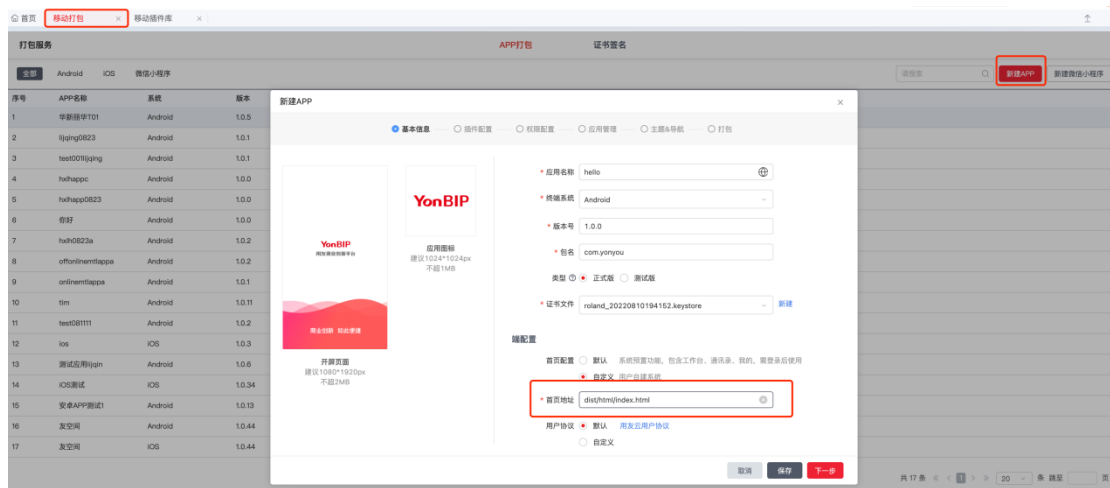


5.3 创建项目

5.3.1 基本信息

导航到：移动工作台-应用构建服务-移动打包，点击新建 APP，需要配置项目的图标，开屏页面，应用名称，系统，初始版本，包名，证书，用户协议，需要注意的是首页配置：选择自定义选项，首页地址要填写离线包中的首页地址，即 3.1 制作的离线包，我们把项目源码都拷贝到了离线包中的 dist 目录下，所有首页地址是从 dist 目录开始。例如：

项目的首页为 html/index.html，那么首页地址配置为：dist/html/index.html。



5.3.2 插件配置

配置完项目基本信息后，下一步到插件配置时，需要我们上传的插件。



添加插件

系统 **自定义**

全部 设备访问 功能扩展 开放SDK 人工智能 直播/音视频 轻应用

插件名称	描述	分类	类型	提供者	更新时间	最新版本
<input type="checkbox"/> hxli_test_003	离线包插件	设备访问	H5	用户lijqing	2022-08-24	1.0.0
<input type="checkbox"/> upesn-plugin-...	引入mtl测试	设备访问	H5	系统插件	2022-08-24	1.0.3
<input type="checkbox"/> hxlh0823c离线包	hxlh0823c离线...	设备访问	H5	just1	2022-08-23	1.0.0
<input type="checkbox"/> hxlh0823b离线...	hxlh0823b离线...	设备访问	H5	just1	2022-08-23	1.0.1
<input type="checkbox"/> hxlh0823a离线包	hxlh0823a离线...	设备访问	H5	just1	2022-08-23	1.0.2
<input type="checkbox"/> offonlinemtlappa	offonlinemtlapp...	设备访问	H5	just1	2022-08-17	1.0.0
<input type="checkbox"/> mtlidemo0811	离线插件形式打包	开放SDK	H5	just1	2022-08-11	9.0.0
<input checked="" type="checkbox"/> mtl-com-custo...	0810自定义离线...	开放SDK	H5	just1	2022-08-10	13.0.0
<input type="checkbox"/> h5-plugin-kenny	工作台离线包	设备访问	H5	系统插件	2022-08-16	3.0.0
<input type="checkbox"/> 高德地图(带界面)	高德地图选择位...	开放SDK	原生	系统插件	2022-07-26	1.0.0

共 10 条 << < 1 > >> 10 条 跳至 页

取消 确定

新建APP

基本信息 **插件配置** 权限配置 应用管理 主题&导航 打包

已选择插件

序号	插件名称	描述	分类	类型	提供者	更新时间	版本	配置参数
1	mtl-com-custo...	0810自定义离线...	开放SDK	H5	just1	2022-08-10	13.0.0	<input type="checkbox"/>

5.3.3 权限配置

插件配置完成后，下一步到权限配置，项目如有读取本地文件，打开照相机等需求，都需要添加相应的权限，点击添加权限，勾选相应的权限。

配置APP的隐私权限

添加权限

权限 • 描述



未设置任何权限

添加权限

已选权限(0)

<input type="checkbox"/>	权限	• 描述
<input type="checkbox"/>	电话	电话
<input type="checkbox"/>	录音	录音
<input type="checkbox"/>	短信	短信
<input type="checkbox"/>	位置	位置
<input type="checkbox"/>	通信录	通信录
<input type="checkbox"/>	摄像头	摄像头
<input type="checkbox"/>	蓝牙	蓝牙
<input type="checkbox"/>	闪光灯	闪光灯
<input type="checkbox"/>	系统日志	系统日志

已选0个

取消

确定

配置APP的隐私权限

添加权限

权限 • 描述

摄像头 摄像头

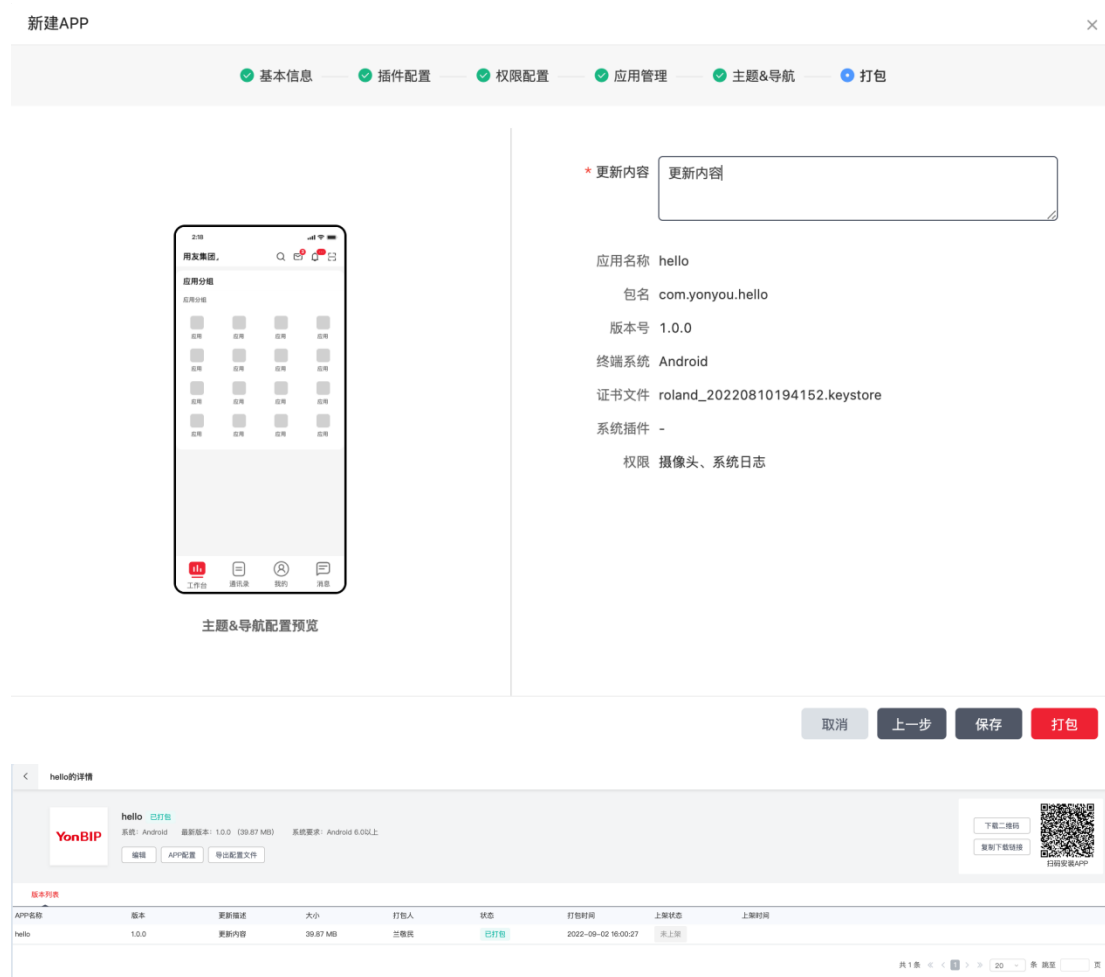
系统日志 系统日志

5.3.4 应用管理 主题&导航

配置完权限后，下一步到应用管理和主题&导航，默认即可。

5.3.5 打包

所有的配置都完成后，到最后一步，打包，输入更新内容，进行打包操作。



附录 迁移优化建议

(1) 如图所示，对项目采用 vue 或者 react 等框架开发的项目来说，属于单页面形式，体验不好，建议采用 apicloud 的 win + frame 方式。



(2) 项目中的网络请求，建议用通过调用 mtl.request 接口进行实现。