2017-12-03APP打包密钥和相关文件申请

H5 发行iOS和 Android 原生应用包

1 . 密钥申请

不管是 ios 和 Android ，申请密钥时都需要 java（java 8） 环境。window 系统也可以申请 ios 密钥和描述文件，需要借用第三方工具 APPuploader （百度可以下载，此工具需要java环境和ios开发者账号）

iOS

首先，打开APPuploader ，点击苹果开发者中心，登录ios开发者账号， 点击 --> Certificates, Identifiers & Profiles ， 进入 证书管理界面，这里我们只需要申请一个 APPID 就行了，其他的东西可以在APPuploader里面操作，点击--> Identifiers --> appids ，新建一个应用，这里就不用赘述了。

appid 申请好之后，回到APPuploader 里面，创建证书和表述文件，这里需要注意的是，ios 应用（证书）有几种类型，如下：

iOS开发证书 ： （真机调试测试用）限制100台设备

iOS发布证书 ： （发布上架APP store）

iOS推送证书 ： （APP推送证书）

iOS企业证书 ： （免上架App Store安装手机使用）无限制

（这里，主要是看 APP 的业务需求，如果只是测试就没必要去花钱申请企业证书这种）。

点击 --> 证书，进入证书页面之后，点击 --> add ，添加证书，这里选择的ios证书类型就是我们需要创建的应用类型。下面的名称，邮箱，和密码自己设置，最好是是自己常用的，能记住就行，OK之后，就会自动生成 p12 文件和证书文件（如果不是使用APPloader的话，我们需要在mac电脑上面用key工具合并request文件和证书文件才能生成p12文件），这个就是我们打包APP的时候需要的证书文件，可以把p12文件下载到本地，打包的时候回用的到。

生成好证书之后，点击back键回到APPuploader主页，点击 --> Profiles（描述文件，这个是跟我们申请的证书文件关联的），进入描述文件页面，点击add ；创建一个描述文件，

文件类型：选择 Inhouse（因为我们创建的证书是企业证书，这里的类型也必须是跟证书对应，不然会提示你没有创建证书）；

应用id：这里的应用ID就是我们之前在ios开发者中心创建好的APPid，如果有多个的话，一定不能选错；

证书：选中应用id 会有一个默认证书；

Device：没有就不选（有也不用选），这里是调试设备；

名称：自定义，自己记住就行；

点击OK，我们的profles文件就创建好了，点击下载，保存到本地（后面打包会用到）；

到此，iOS 证书和描述文件就申请完了；

Android

Android 相对ios就比较简单了，在本地运行命令行，键入：keytool -genkey -alias android.keystore -keyalg RSA -validity 36500 -keystore c:\key\android.keystroe;

-genkey ： 生成文件；

-alias ： 证书别名（打包时候会需要）；

-keyalg ： 加密算法；

-validity ： 有效期；

-keystore ： 文件名（生成的文件打包的时候会需要）；

然后填写依次填写密钥库口令、确认口令、姓名与姓氏、组织单位、城市或区域、省/市/自治区、国家/地区代码，最后如果正确的话填“y”，如果错误的话直接“Enter”下去；（这里一定要注意的是，如果c盘下面没有设置的目录，就会报错，还有，后面完成的时候，一定要y同意一下）；

然以，进到刚刚保存的key目录下面，启动命令行，键入 ： keytool -list -keystore "android.keystore"；就可以看到证书相关的信息；

到此Android证书也申请好了；

2 . 第三方应用申请

我们的APP里面，有可能会用到推送、第三方登录、统计、支付等功能，这里我们就需要在第三方应用的平台去创建开发者账号，并且申请APP需要的 appid、appley、appsecret 等密钥；将第三方应用接入到APP里面；拿个推为例。

第三方推送

首先，我们需要在个推的官方网站上，注册一个开发者账号（有的第三方应用是需要花钱创建的，例如：微信支付），登入到开发者中心，这里有给我们提供一个个推的demo，这里我们可以忽略，点击登记应用，需要我们填一个表单：

应用图标：这个就不用说了，让ui提供；

应用名称：这个就是我们需要接入的APP名称；

应用类型：随便选；

应用平台：Android为例（iOS 有两个选项，我们要选择iOS 生产环境）；

应用标识：就是我们打包APP的包名，这个一定不能填错 （例如：com.76666a.caishijie）；

点击确定，回到应用列表里面，就可以看到我们刚刚创建好的应用了；

看到我们刚刚创建好的APP之后，点击应用配置，可以查看当前应用的相关配置：

应用信息：图片、类型、名称；这一栏也没什么可以说的，就是刚刚配置好的（可以修改）；

应用配置：这里就可以看到，个推平台给我们的应用的接口，APPID、AppSecret 、AppKey 、MasterSecret等信息

应用标识：就是我们的应用的包名。可以修改

在使用个推功能之前，我们还需要在我们的APP，manifest.json 文件里面做一些相关的配置，点击SDK配置项，可以看到，推送模块里面有个推和小米推送，这里我们把刚刚申请好的APPID和相关的其他信息填到配置文件里面。保存，重新打包运行，就OK了。（注意：这里必须是打包运行，真机调试是没有用的，包括后面用到的其他的第三方模块都是需要打包运行的）；

在手机上安装好打的apk包之后，可以的登入个推平台推送一条消息试试（如何推送，这里也不做赘述）。

3 ． APP打包和更新版本

打包的时候，焦点一定要在需要打包的项目的index页面！！！

点击发行为原生APP安装包，这里需要填的证书别名、私匙密码、证书文件，都是我们之前申请好的。填完之后点击确定就行，打包成功之后，会自动下载安装包，下载完成之后，本地的项目目录里面会多出一个unpackage --> release文件夹下面（多了一个apk的文件），这个文件就是手机的安装包；

发布更新功能：

APP发布更新，需要服务器配置，服务器目录下面有Android 和 iOS 文件夹以及update文件夹（此文件夹就是用来存放更新版本的json文件），更新文件格式如下：

```
{
    "appid":"H5F6C413B",
    "iOS":{
        "version":"iOS新版本号，如：1.0.0",
        "note":"iOS新版本描述信息，多行使用\n分割",
        "url":"Appstore路径，如：itms-apps://itunes.apple.com/cn/app/hello-h5+/id682111190?l=zh&mt=8"
    },
    "Android":{
        "version":"1.0.1",                                          //每次更新的版本号
        "note":"新的测试版本！\n 各种真人游戏，电子游艺，等你来玩！",          //跟新版本客户端收到的更新消息
        "url":"http://res.am01.com/android/caishijie-android.apk"   //每次更新包的下载地址
    }
}
```

也就是说，每次打包发行APP ，必须要拷贝两个文件到服务器上，一个是打包好的apk文件，一个是update.json文件，这里的版本号必须要与打包配置的版本号保持一致；

4 ． 客户端id （CID）获取

推送服务需要用到一个 clientid ，这个是跟推送的 appkey 相关联的，在我们的项目里面集成个推之后，就可以直接调用mui 的plus下面的方法获取clientid ，具体代码如下：

```
<script type="text/javascript">
    mui.init();
    mui.plusReady(function(){
        var info = plus.push.getClientInfo();
        console.log("token: "+info.token);
        console.log("clientid: "+info.clientid);
        console.log("appid: "+info.appid);
        console.log("appkey: "+info.appkey);
    })
</script>
```

这是真机测试的获取方法，有可能不真实，需要打包后重新获取。

5 . ios 安装包扫码下载

苹果app包下载安装，如果直接下载 ipa 是不能在手机上安装的，必须要通过 （ itms-services://?action=download-manifest&url=https://www.domain.com/app.plist）这种链接下载（绕过store下载安装），url 前面是固定的写法，后面的域名必须是https的，这里，我们需要把我们的服务器加上ssl认证，然后在服务器上部署一个后缀名为 .plist 的文件（xml格式的），下载链接是先访问这个plist文件，在文件里面配置我们的 ipa 下载地址；plist 文件配置如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0.0">
<dict>
    <key>items</key>
    <array>
        <dict>
            <key>assets</key>
            <array>
                <dict>
                    <key>kind</key>
                    <string>software-package</string>
                    <key>url</key>
                    <string>http://app76660.com/test/ios/caishijie-ios.ipa</string>        //ipa下载地址
                </dict>
                <dict>
                    <key>kind</key>
                    <string>display-image</string>
                    <key>needs-shine</key>
                    <true/>
                    <key>url</key>
                    <string>http://app76660.com/test/ios/icon.png</string>         //ipa下载时候logo显示图片
                </dict>
            </array>
            <key>metadata</key>
            <dict>
                <key>bundle-identifier</key>
                <string>io.dcloud.H5F6C413B</string>            //应用id
                <key>bundle-version</key>
                <string>1.0.0</string>                                  //应用版本号
                <key>kind</key>
                <string>software</string>
                <key>title</key>
                <string>福利彩世界</string>              //应用名称
            </dict>
        </dict>
    </array>
</dict>
</plist>
```

6 . 第三方服务平台跳转

APP里面有使用第三方链接，不能像浏览器那样直接 window.open（），因为这样打开的网页是基于 app 打开的，是app里面集成的 navigater 环境，集成环境的浏览器兼容性是非常差的；

原来的方案是直接在原有的页面里面嵌入 iframe ，结果是有的第三方平台之间跳转了很多次，最后跳转过去的还是直接 window.open()打开的页面，不在我们能控制的范围之内；

最终解决方案，是在我们自己的服务端部署一个静态页面，前端将接口读到的参数通过 url 传到服务端的静态页上面，由这个页面发起请求，这样，我们就可以直接调用 手机原生的浏览器去读取服务端的页面，这样做的好处是通过浏览器打开的页面，不管中间跳转了多少次，我们的app进程还在，也解决了之前 app内 打开杀掉进程的问题；