untitled

《APICloud 实验手册》

Java课程组

版本1.0

文档提供：Java课程组

# 第1章 APICloud开发环境搭建

## 1.1 实验一 SublimeText3环境搭建

### 1.1.1 实验目的

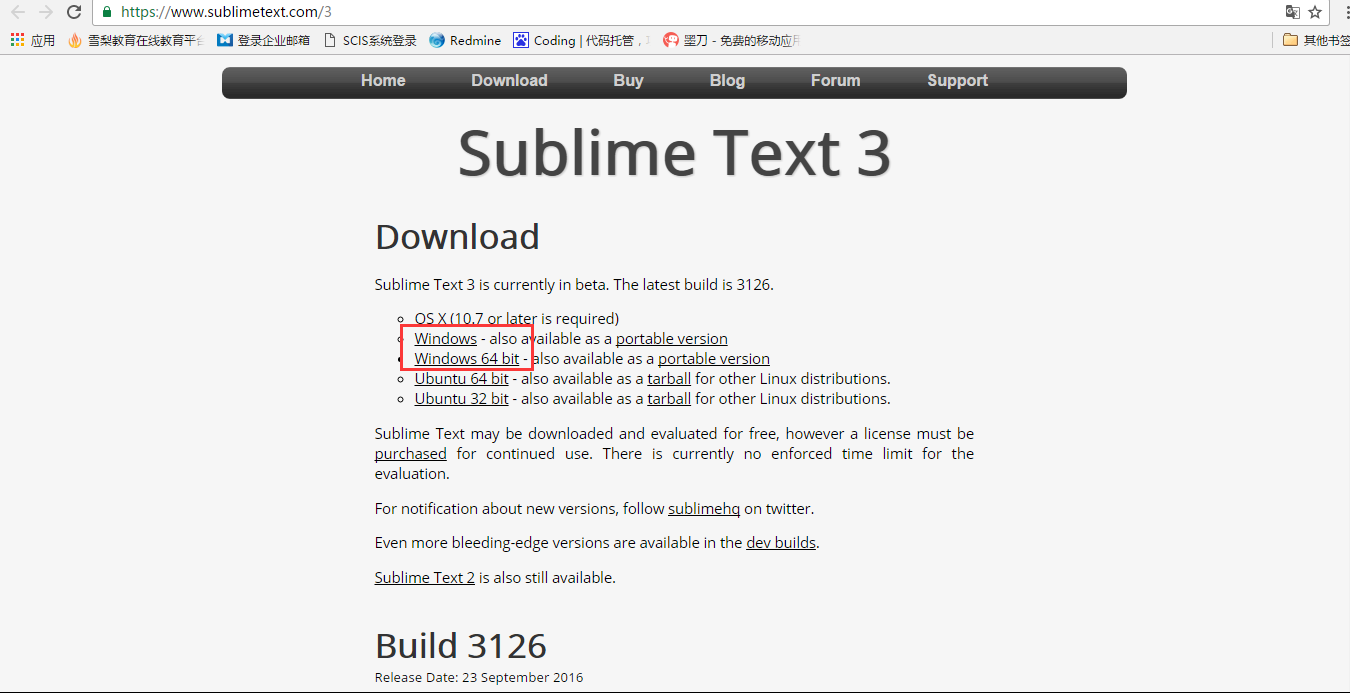
1. 完成APICloud的Sublime Text3开发环境搭建。

### 1.1.2 准备工作

无

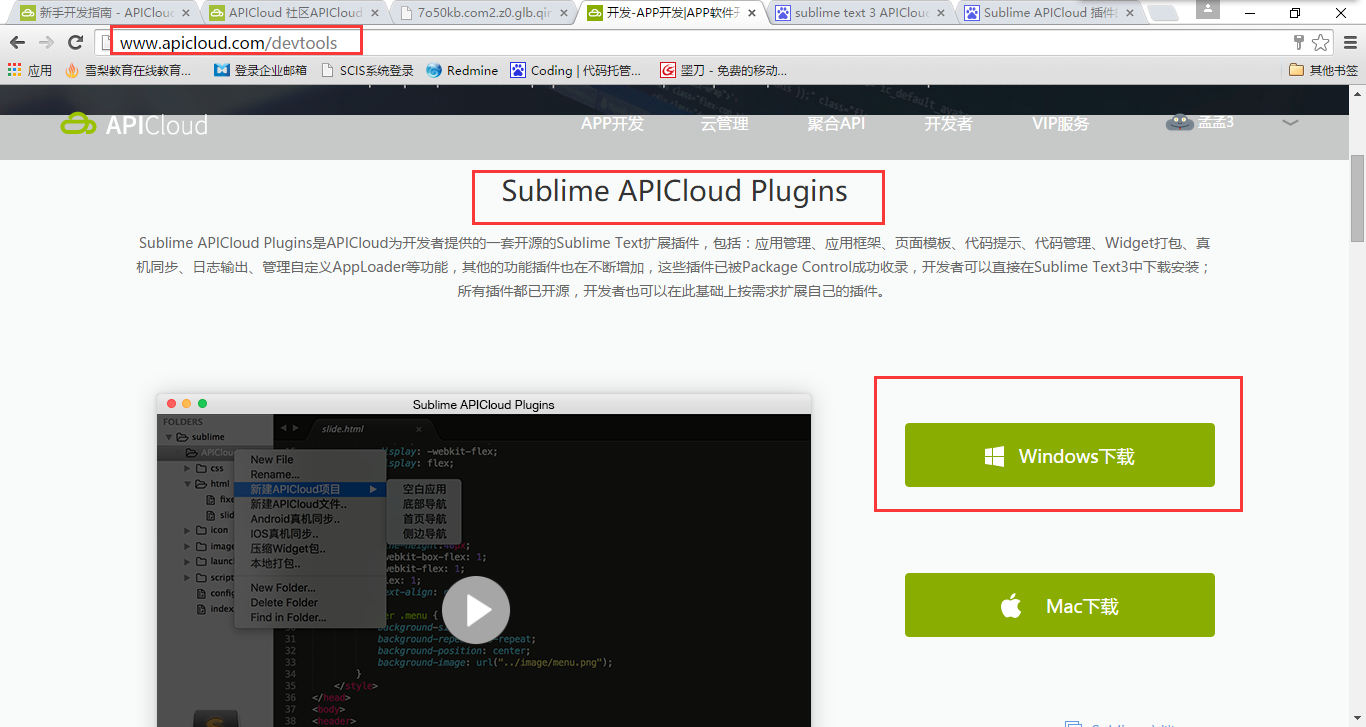
### 1.1.3 实验步骤

1. 从官网下载Sublime Text3安装文件，如下图所示：



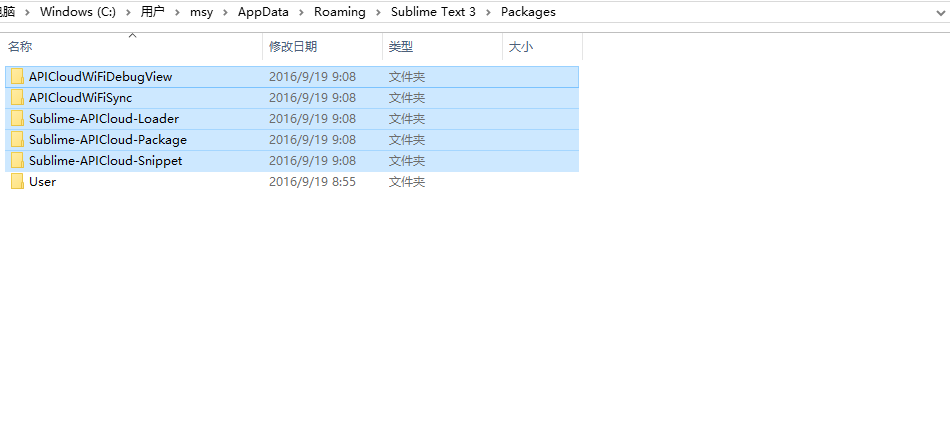
*图 1.1.1*

1. 从APICloud官网下载Sublime APICloud Plugins（Sublime Text的APICloud插件），注意下载的版本，如下图所示：

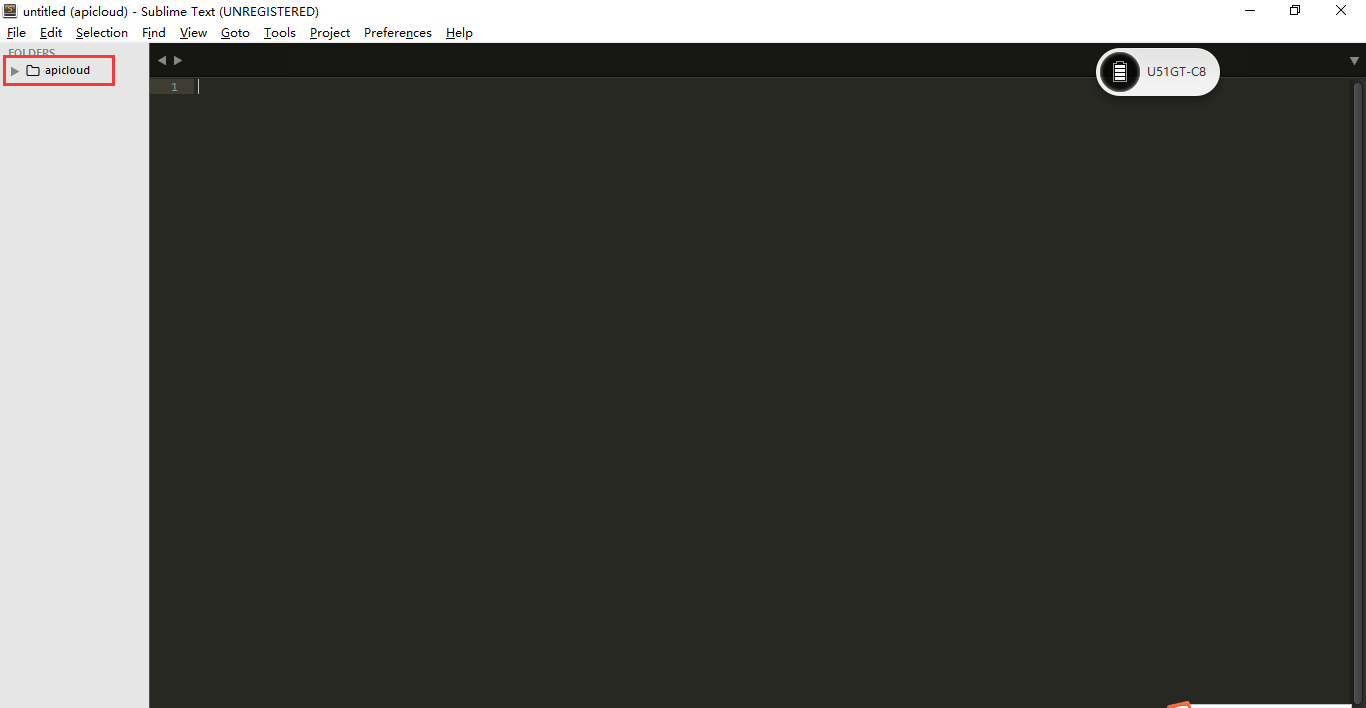


*图 1.1.2*

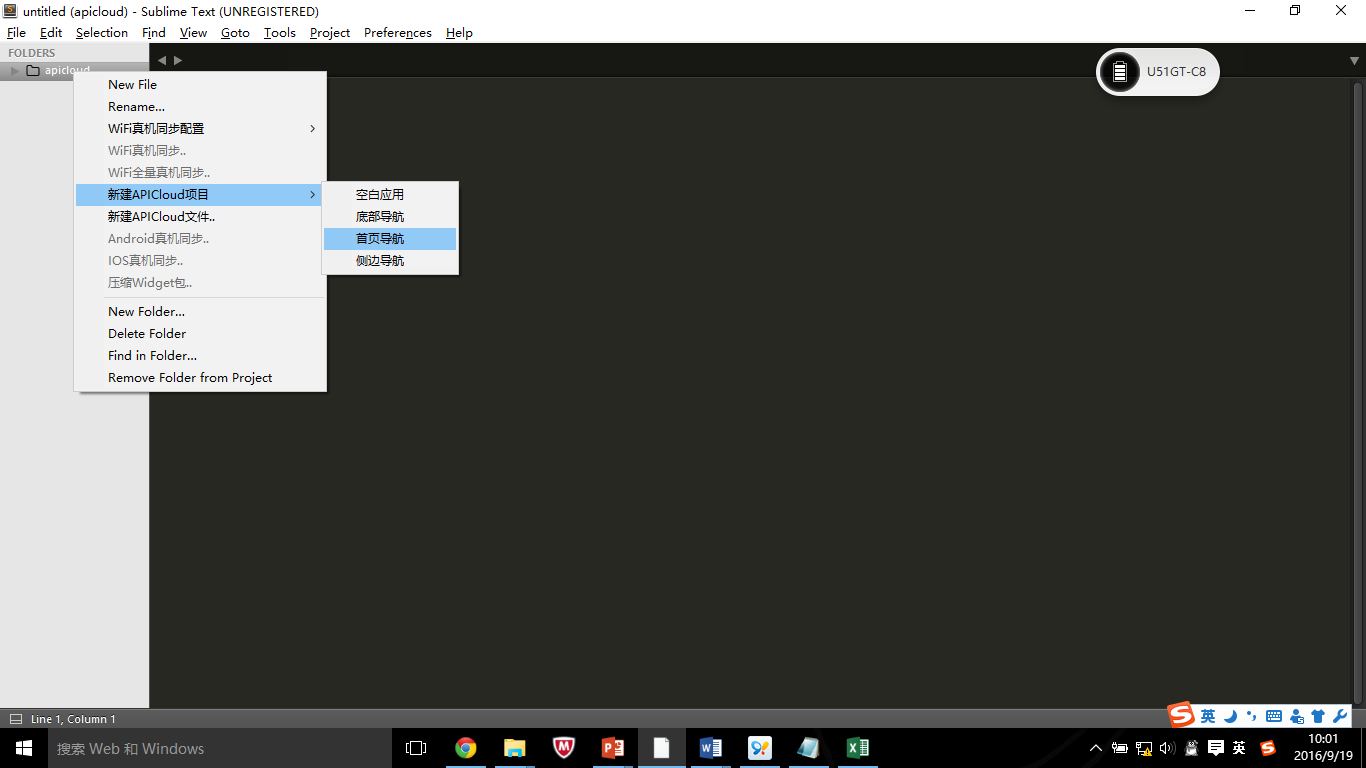
1. 将Sublime APICloud Plugins安装到Sublime Text3工具中，直接将Sublime APICloud Plugins解压后的文件放到Sublime Text3的Packages目录中即可，如下图所示：

*图 1.1.3*

1. 新建文件夹，并用Sublime Text3打开，文件夹的名字自己定义，如下图所示：

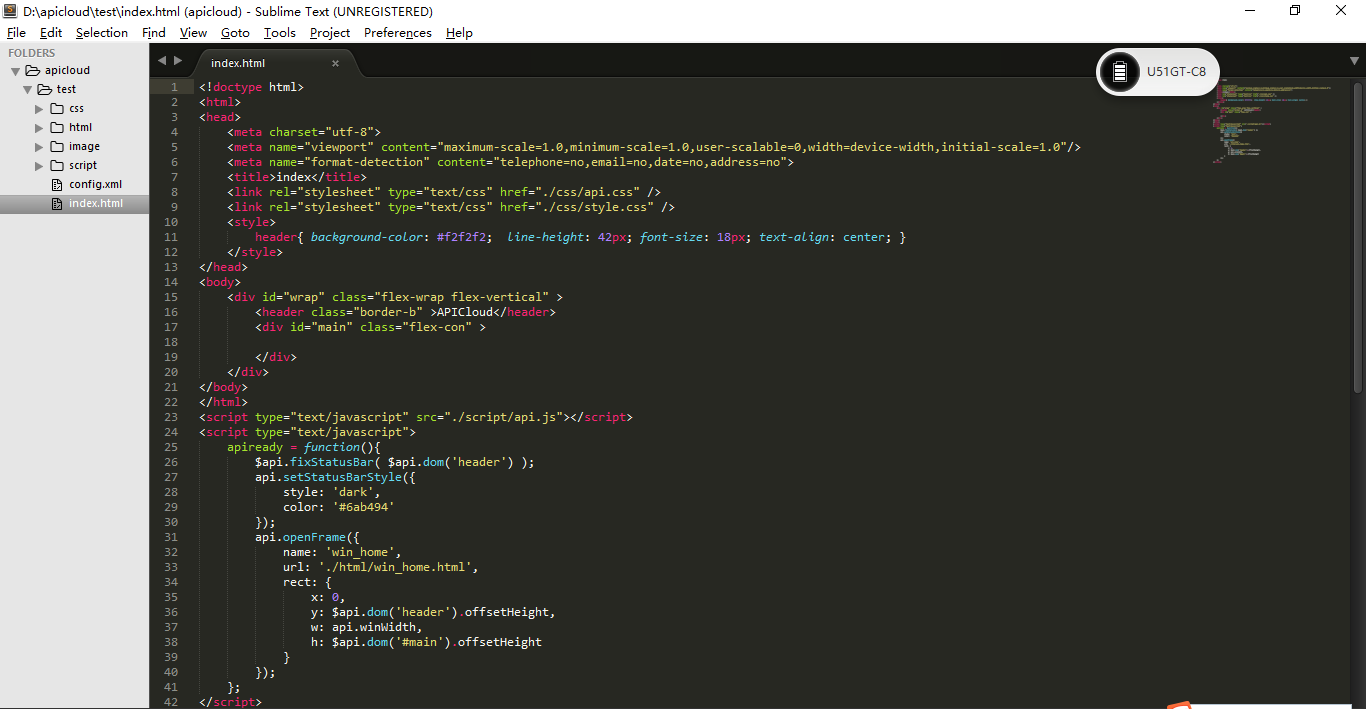


*图 1.1.4*

1. 在APICloud文件夹上右键选择新建APICloud项目，如下图所示。

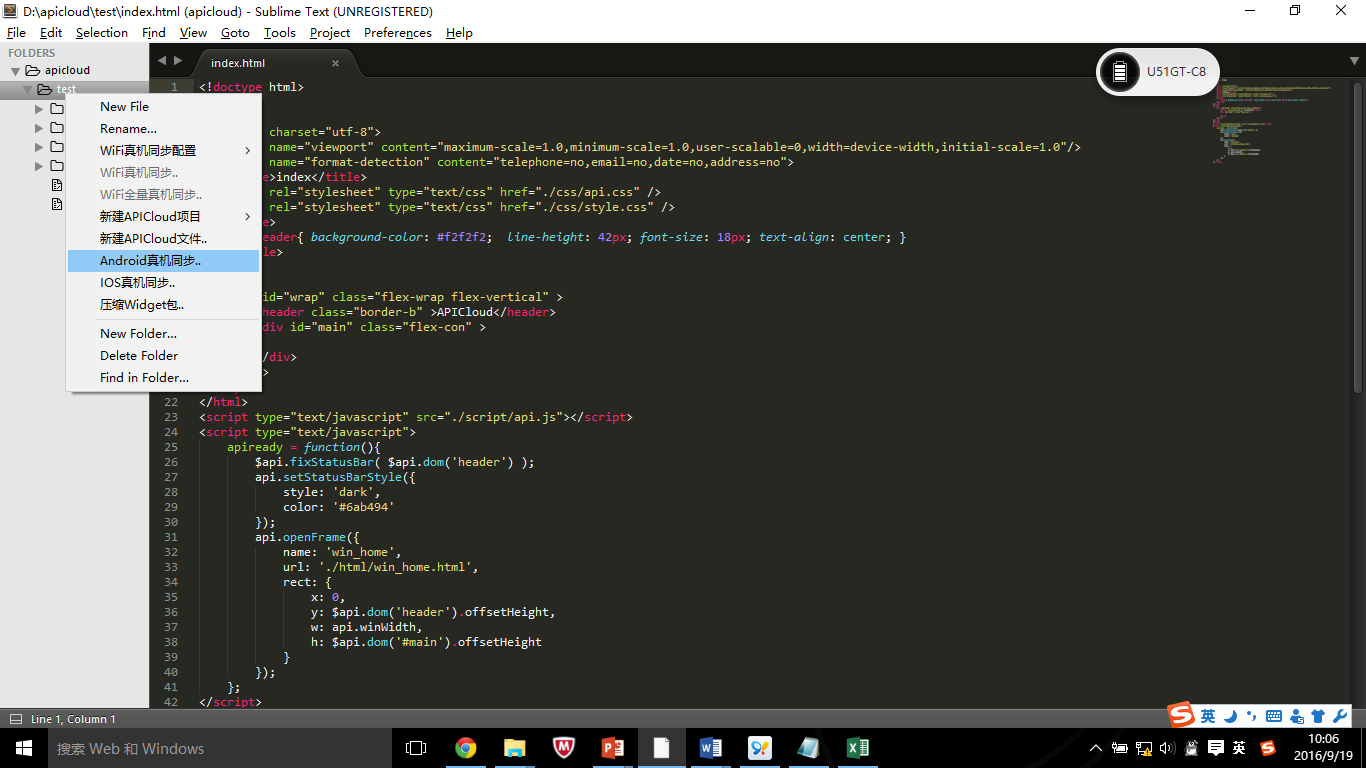
*图 1.1.5*

1. 底部弹出一个输入框 -> 填入 APICloud 项目名称 -> 点击回车键，新建项目创建成功，如下图所示。



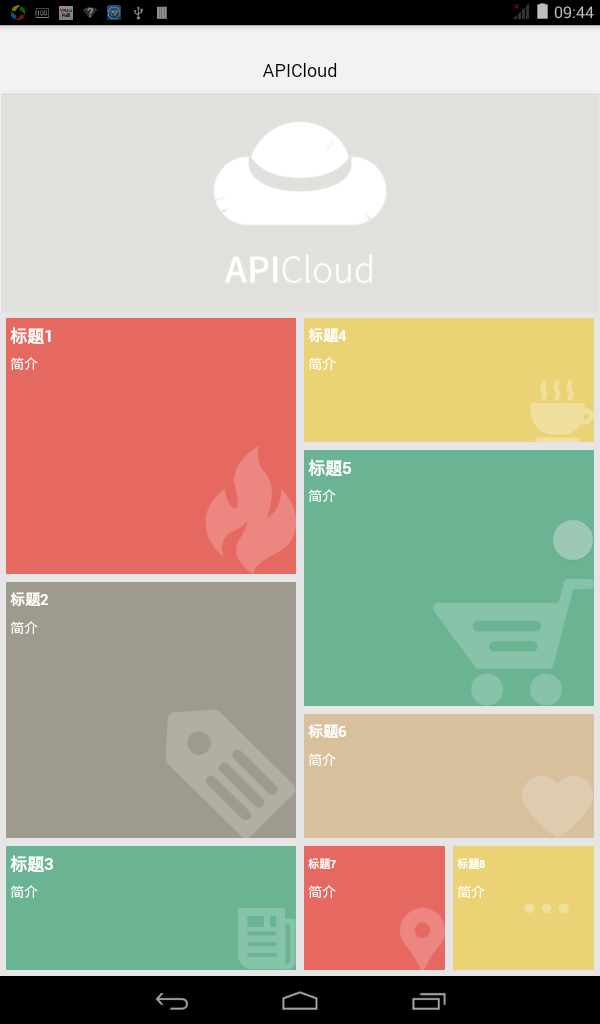
*图 1.1.6 新建test APICloud项目目录结构*

1. 项目的调试和运行。项目可以运行在手机的模拟器中，也可以运行在真机中，在本例中我们选择Android真机设备调试运行，先将手机连接到电脑上，在test项目中点击右键选择Android真机同步，如下图所示：



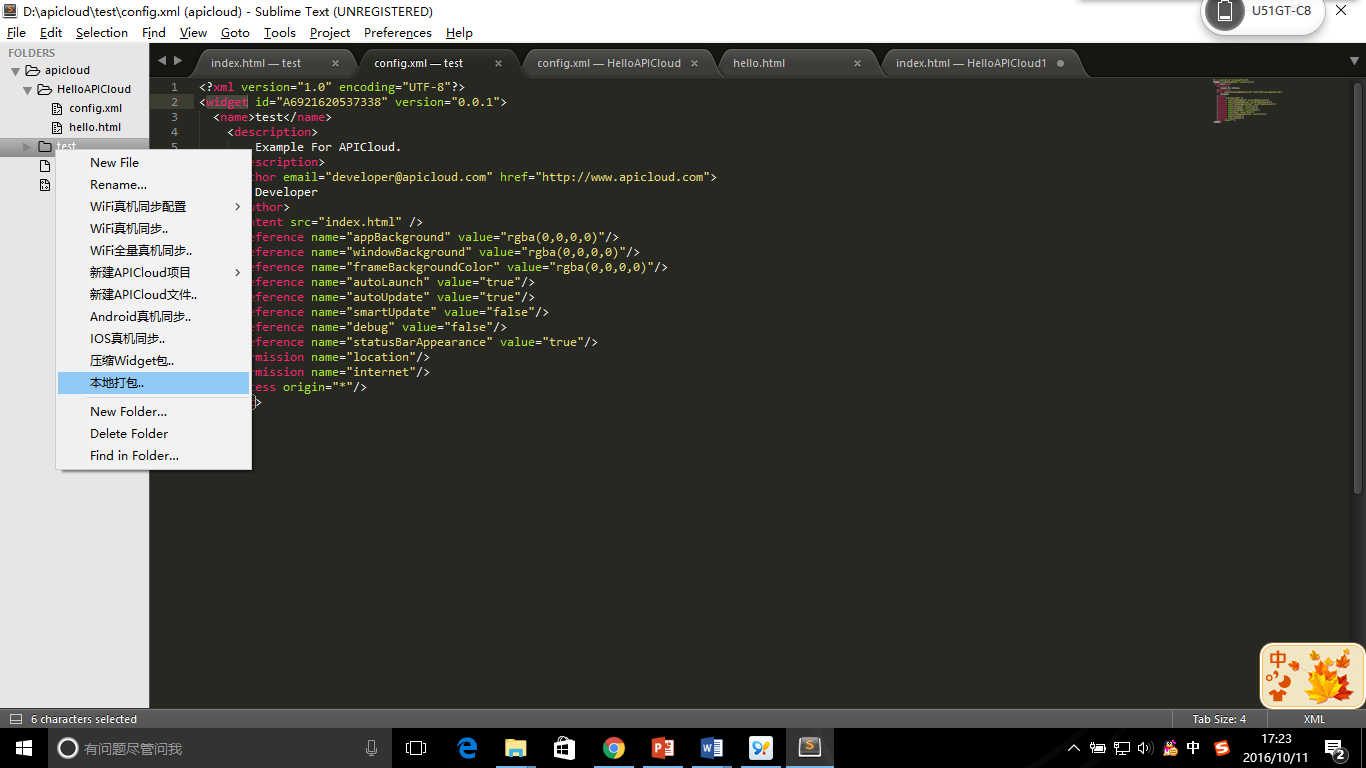
*图 1.1.7*

1. 等待 Android 手机自动打开刚同步的应用，代表同步成功。手机中运行项目的效果如下图所示：



*图 1.1.8*

1. 生成独立的安装文件，在步骤八中生成的效果只是借助APPLoader工具在手机中测试和调试App应用程序，如果想生成独立的安装文件可以在Sublime Text3环境下直接选择工程文件夹，右键选择本地打包。如下图所示：



*图1.1.9 Sublime Text工具生成安装文件*

### 1.1.4 实验结论

成功地配置完成Sublime Text3的APICloud开发环境，并在开发环境中成功地创建并运行第一个APICloud项目test。

## 1.2 实验二 APICloud Studio环境搭建

### 1.2.1 实验目的

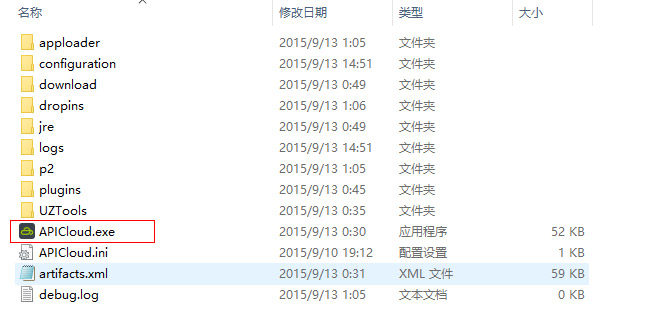
1. 完成APICloud的APICloud Studio 官方开发环境搭建。

### 1.2.2 准备工作

下载适合自己操作系统的APICloud Studio安装文件，下载地址是：[http://www.apicloud.com/devtools#eclipse](http://www.apicloud.com/devtools" \l "eclipse)。

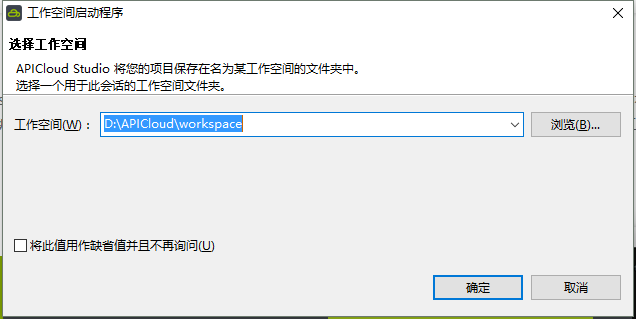
### 1.2.3 实验步骤

1. 启动APICloud Studio，APICloud Studio是绿色版，不用修改注册表。下载后解压缩，运行可执行文件即可。



*图1.2.1*

1. 运行APICloud.exe程序，选择工作空间，以及输入APICloud平台的用户名和密码登录，即可进入到APICloud Studio的主界面，如下图所示：



*图 1.2.2*



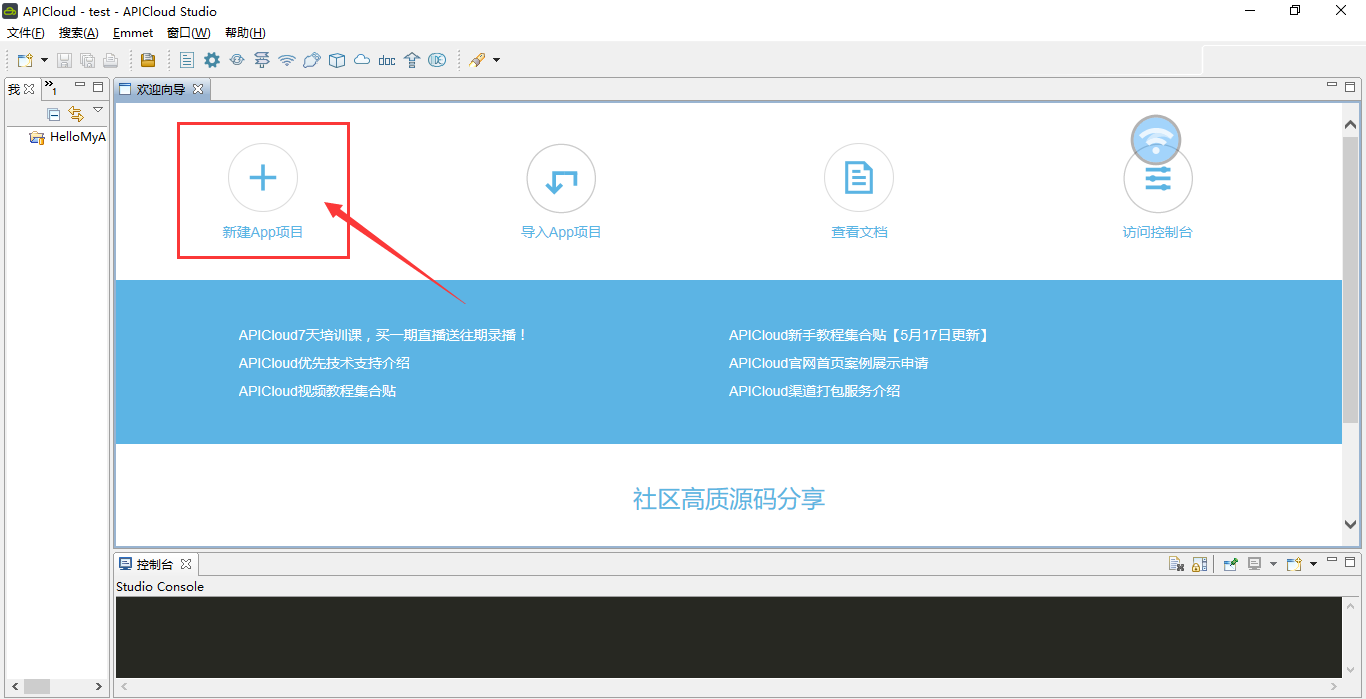
*图 1.2.3*

1. 打开向导页，

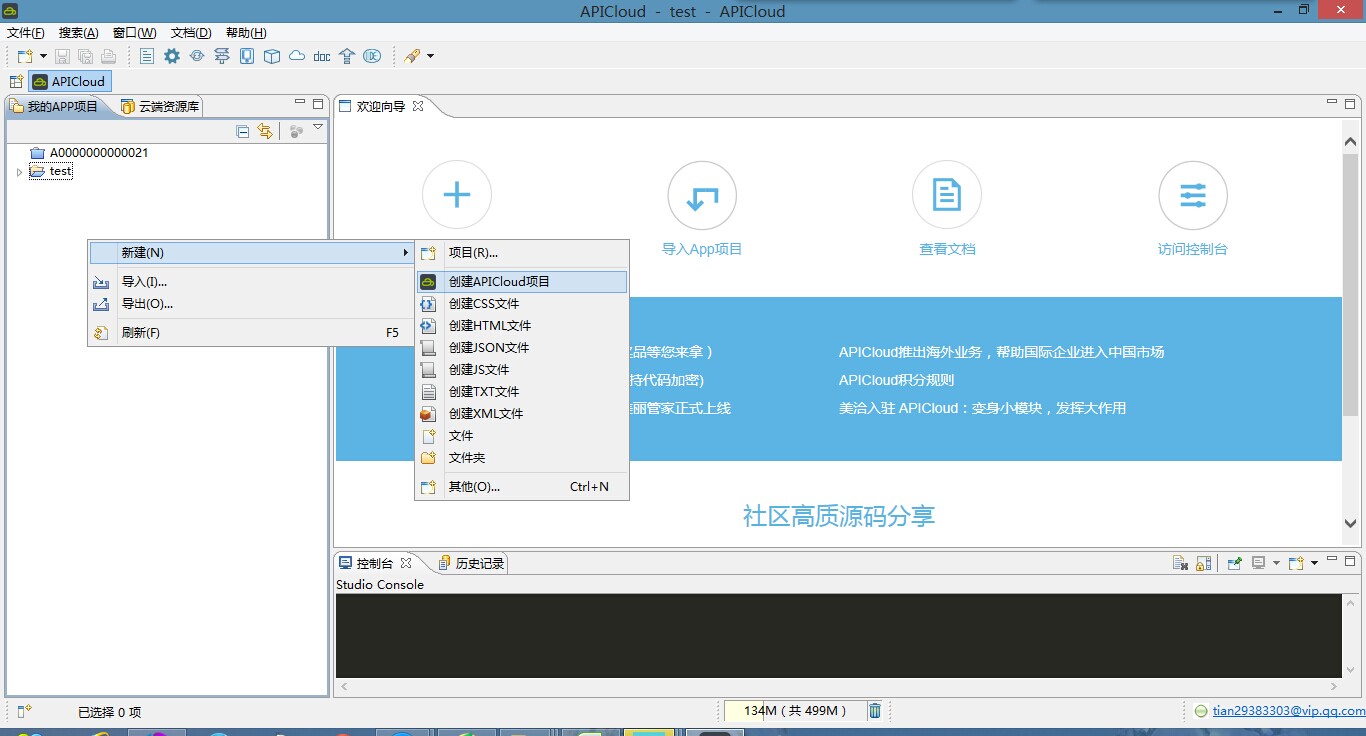


*图1.2.4*

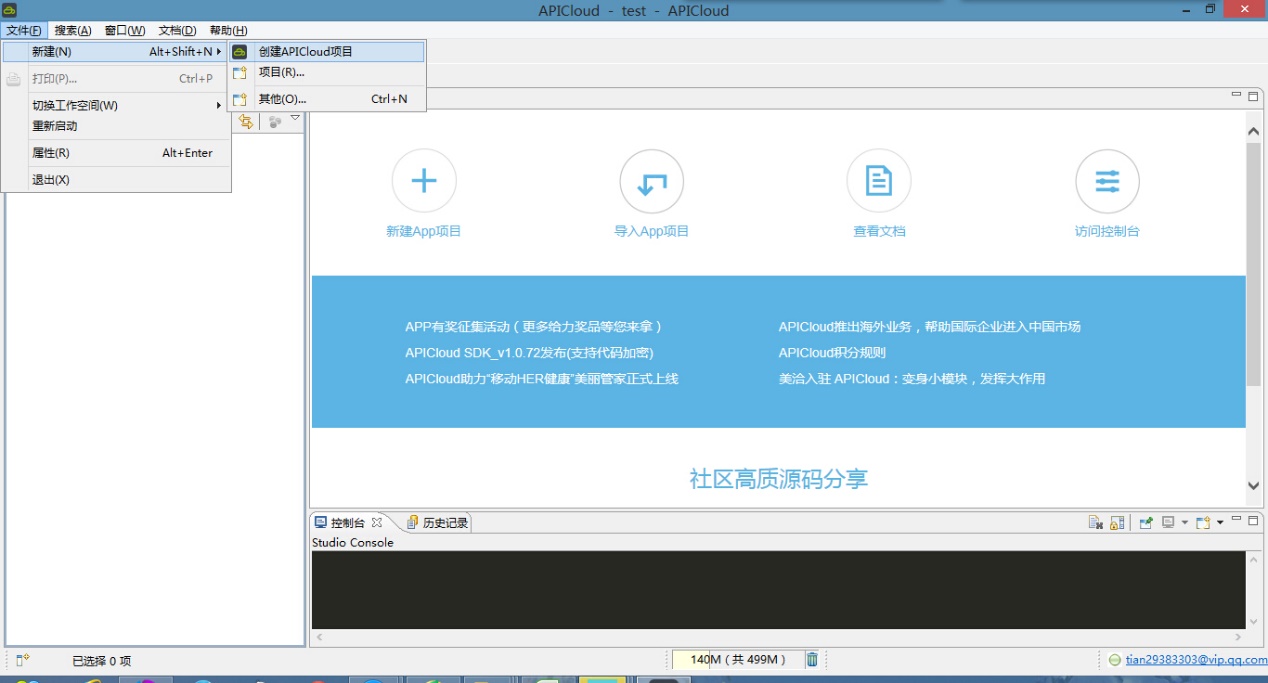
1. 创建第一个移动APP，有三个入口创建，第一：在向导页面中“创建app项目”中创建移动应用；第二：在我的App项目视图中，右键鼠标选择，新建—创建APICloud项目；第三：在文件菜单栏中选择，文件--新建—创建APICloud项目。



*图1.2.5 向导页创建新App*



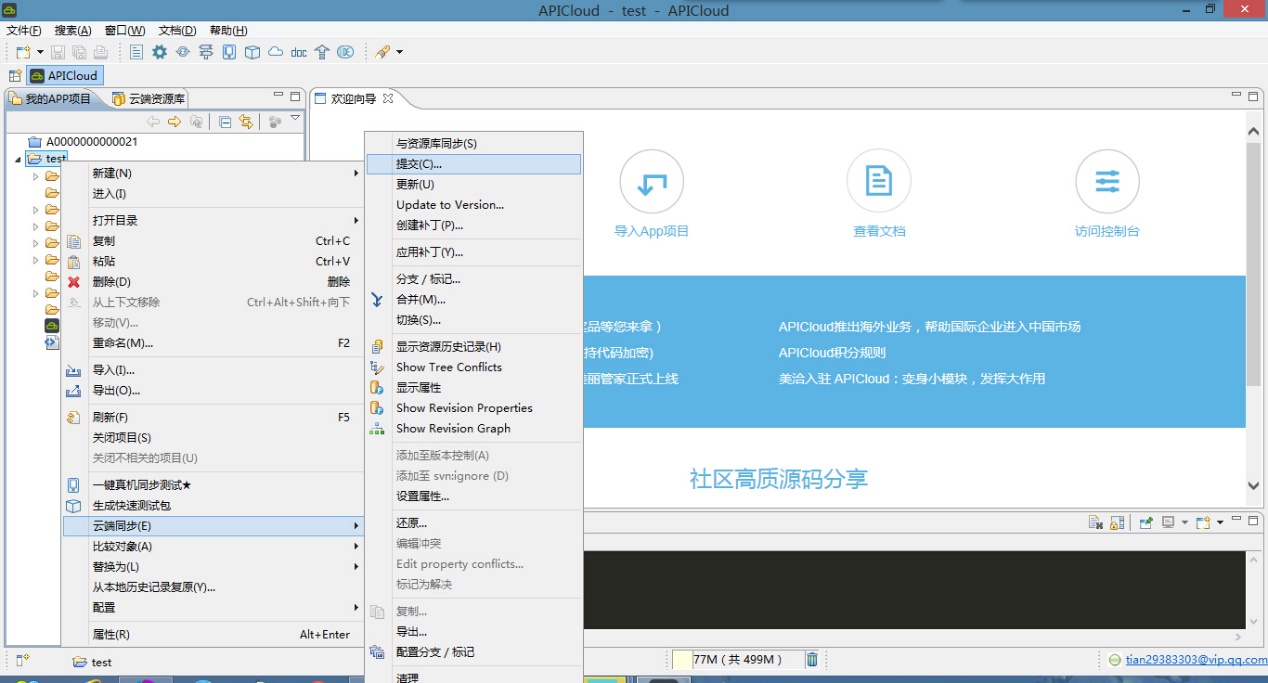
*图1.2.6 APP项目视图创建APP*



*图1.2.7 菜单栏创建App*

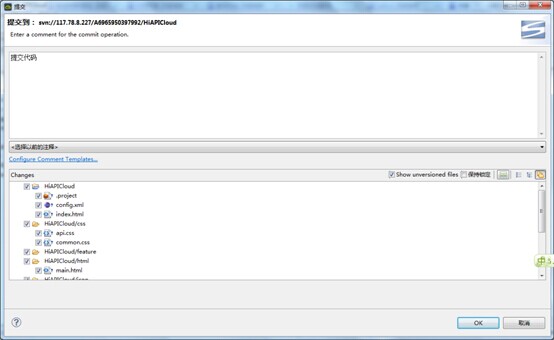
1. 同步本地应用到云端资源库，APICloud Studio开发工具提供了云端上传应用的功能。开发者创建的应用会和云端资源库建立连接。并且网站也通过云端资源库和APICloud Studio共享应用同步开发。

首先选择一个需要同步到云端资源库的APICloud应用。在APICloud应用上右键，选择云端同步—提交。



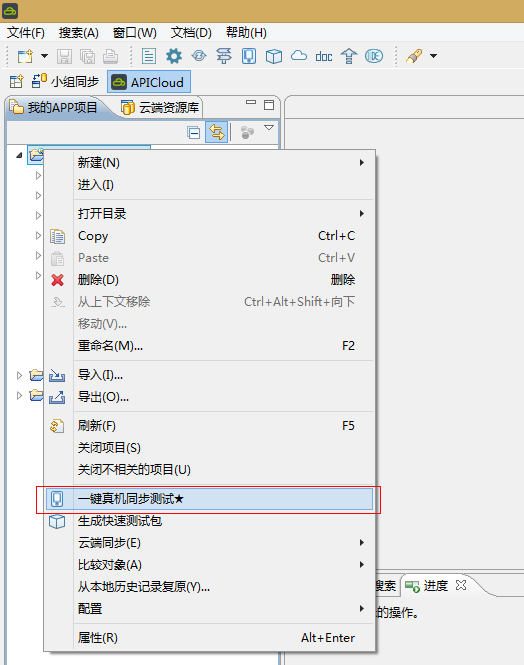
*图1.2.8*

在打开的提交界面输入提交信息后点击完成。即可提交应用到云端资源库中。

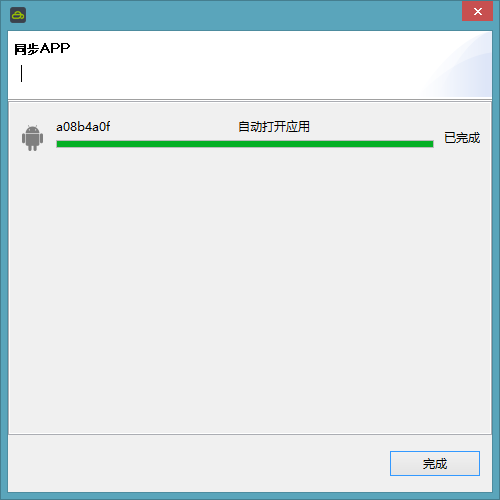


*图1.2.9*

1. 真机同步测试，打开APICloud Studio，用数据线连接移动设备，当前项目下，右键选择“一键真机同步测试”。

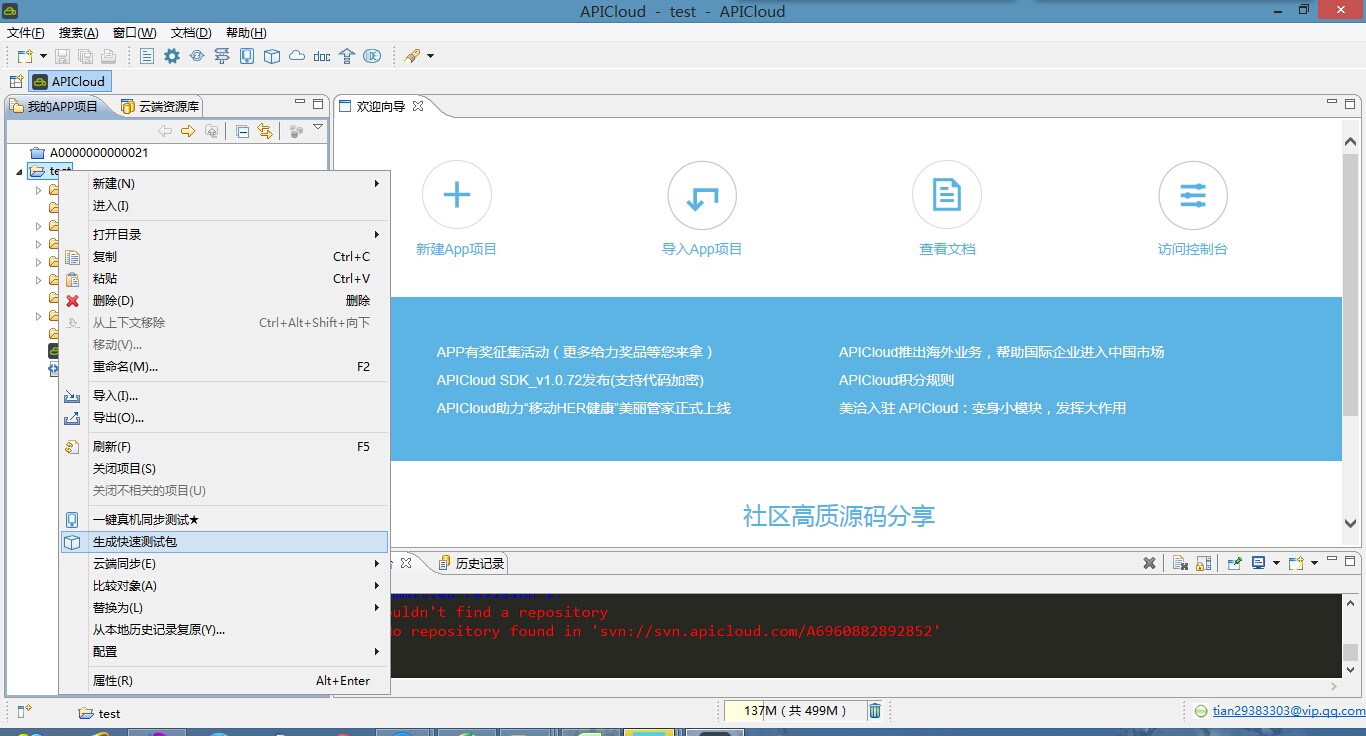


*图1.2.10*



*图1.2.11*

1. 生成独立的安装文件，选择应用项目，右键选择“生成快速测试包”，填写“应用名称”，选择“生成平台”，点击“打包”，即可生成测试安装包。



*图1.2.12*

### 1.2.4 实验结论

成功在本地安装APICloud Studio开发环境，并完成第一个APP程序的测试。

## 1.3 实验三 APICloud云端编译

### 1.3.1 实验目的

1. 借助云端编译生成独立的安装文件。

### 1.3.2 准备工作

在实验一，实验二的基础上进行，即在云端资源库中已经存在了对应的App源程序。

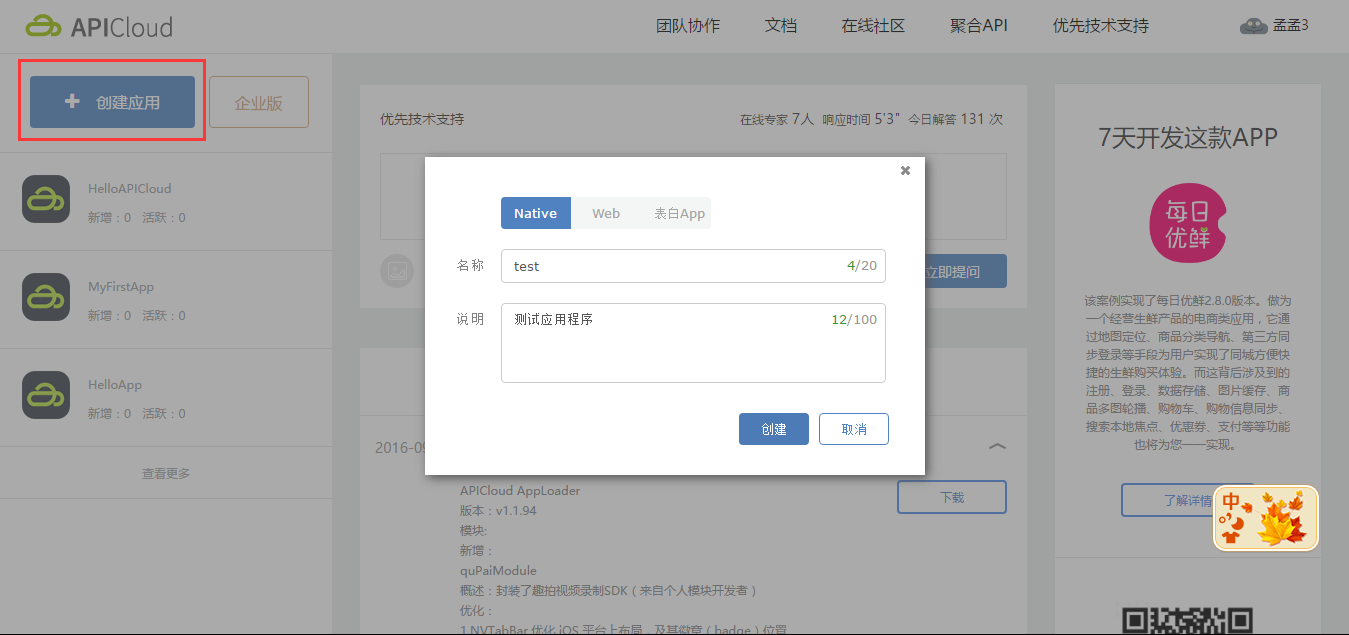
以实验一为例：在实验一的基础上将实验代码同步到APICloud资源库中的步骤如下：

1. 打开APICloud开发控制台首页面，如下图所示：



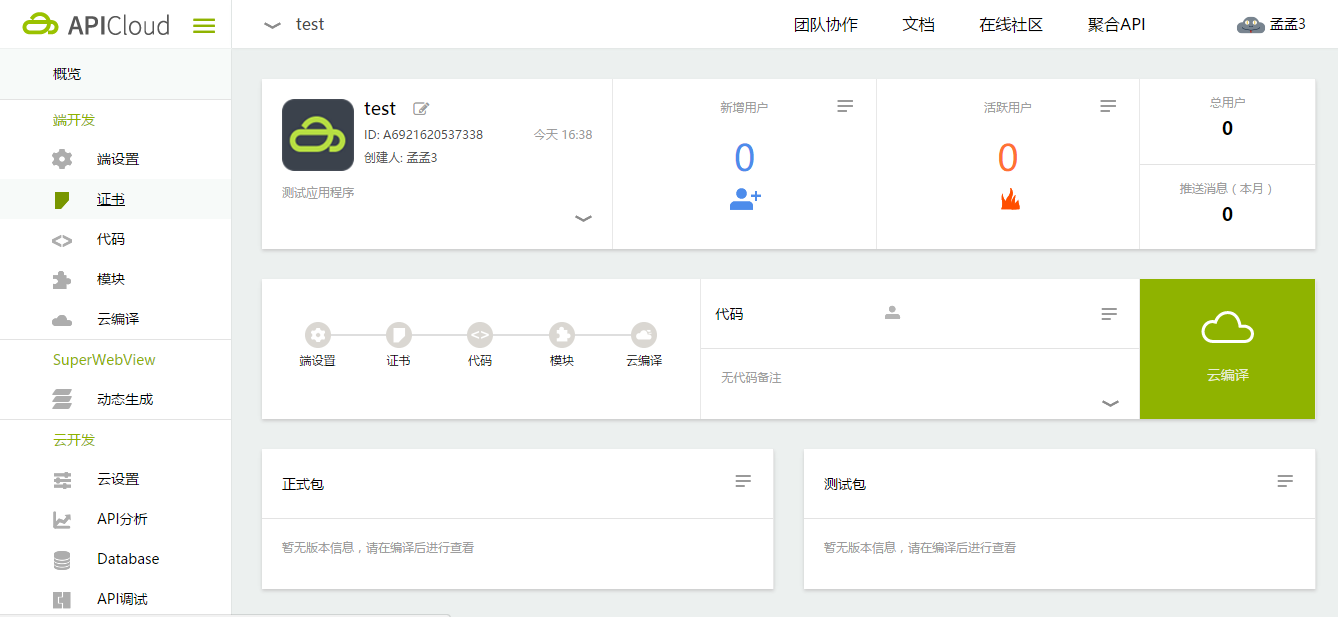
*图1.3.1 APICloud控制台首页面*

1. 创建新的应用，如下图所示：



*图1.3.2 创建APICloud应用的界面*

填写应用的名称和说明，名称尽量与实验一中创建的应用名称一致，将生成的项目编号A6921620537338复制到test源码中config.xml文件中的widget标签的ID属性。

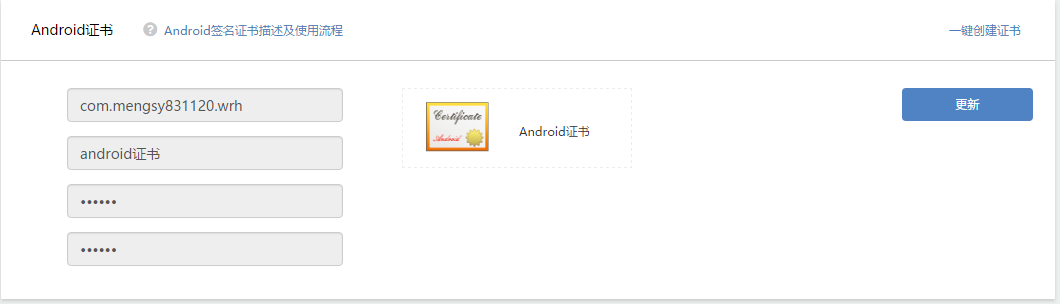


*图1.3.3 创建APICloud应用后的概览界面*

1. 在概览界面中选择端开发模块下的证书项，如果还有没有Android证书，选择一键创建证书，创建一个新的Android证书，如下图所示：

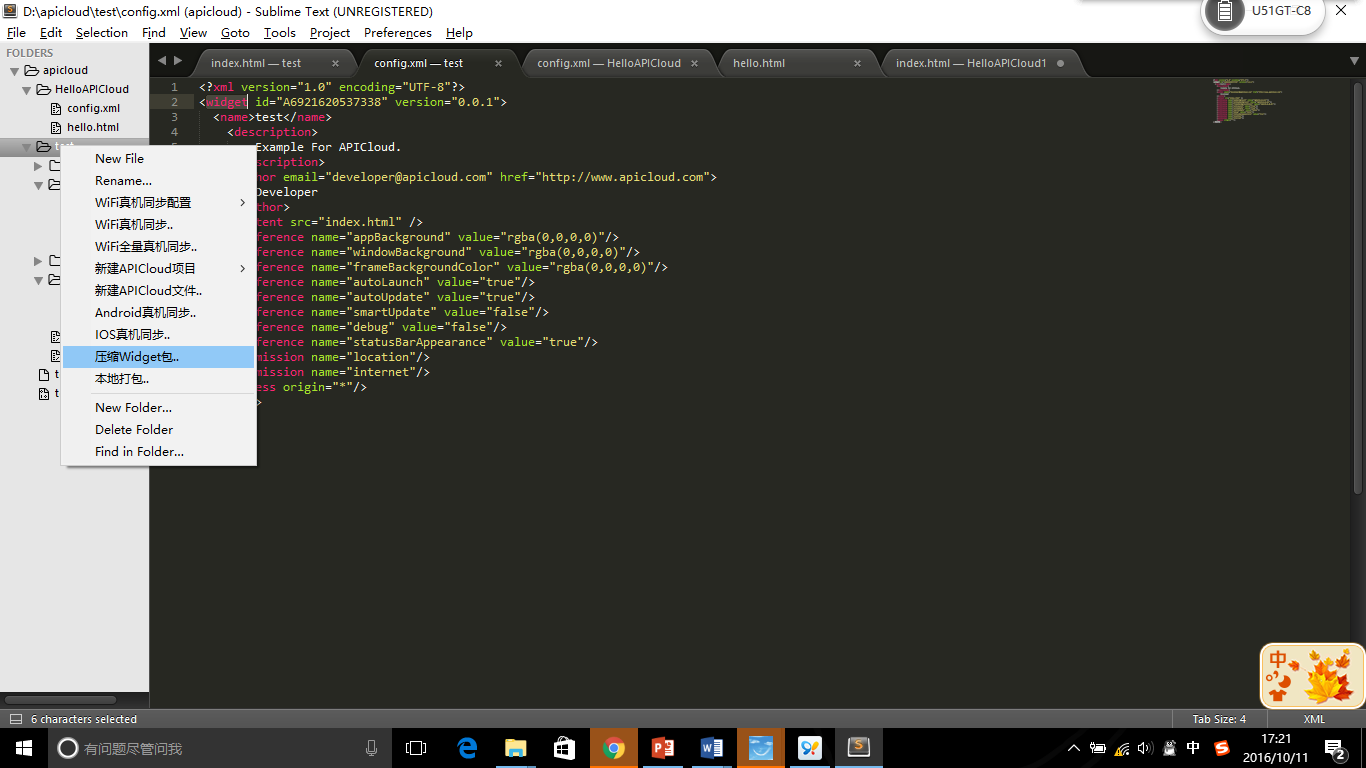


*图1.3.4 一键创建证书*



*图1.3.5 创建证书成功*

1. 将实验一中调试正常的代码打包，如下图所示：



*图1.3.6 代码打包*

1. 在APICloud平台的代码项下，上传打包后的代码，如下图所示：

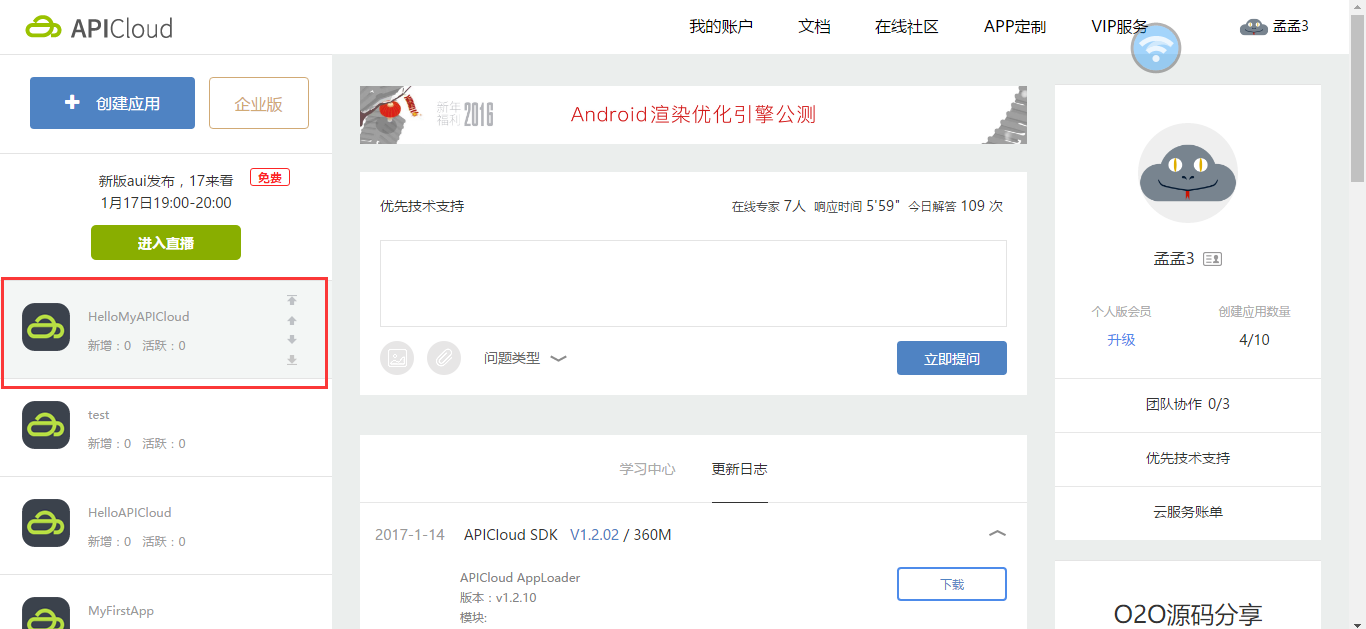


*图1.3.7 上传项目代码*

以实验二为基础，则直接在APICloud Studio工具中同步本地应用到云端资源库即可（即实验二的步骤四）。

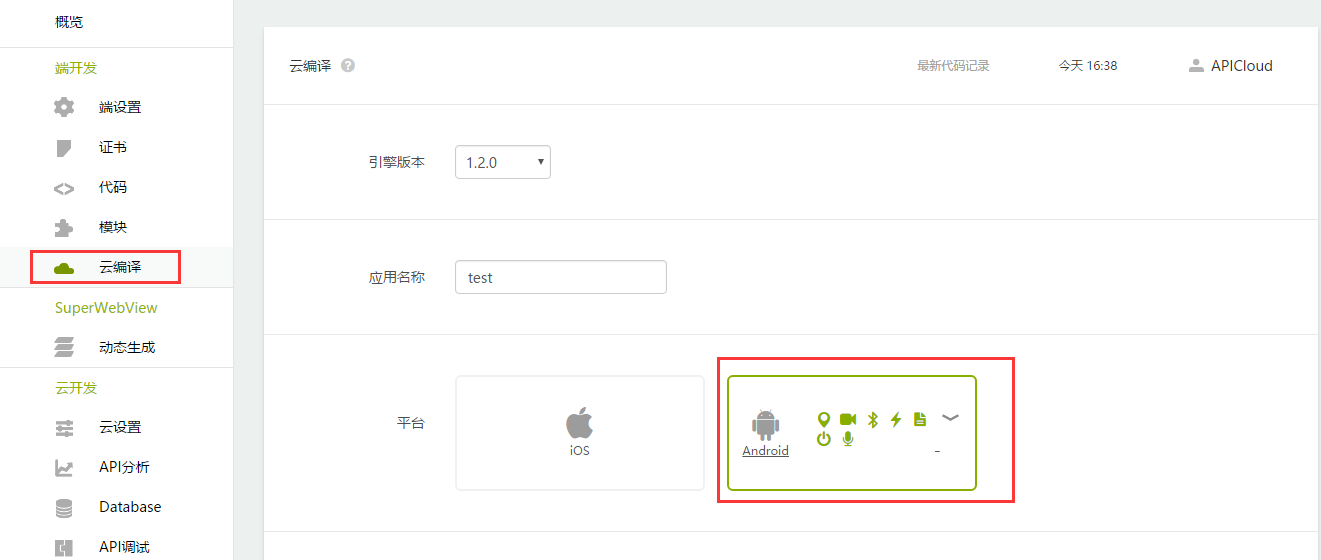
### 1.3.3 实验步骤

1. 打开APICloud开发控制台首页面，选择需要编译的App项目。



*图1.3.8 选择云编译的App*

1. 选择云编译项，勾选编译的选项，如下图所示：



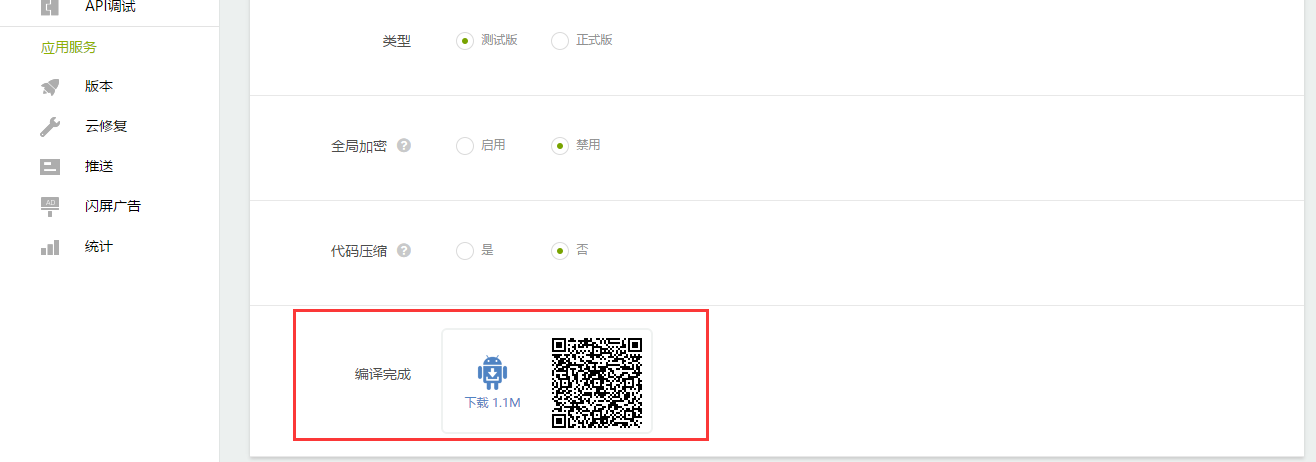
*图1.3.9 编译选项设置*

1. 点击云编辑按钮，开始编译项目



*图1.3.10 编译*

编译成功后如下图所示：



*图1.3.11 云编译成功*

1. 下载Android下的安装文件或者使用Android手机的扫码功能扫描图中的二维码，即可安装此应用。

### 1.2.4 实验结论

本实验主要完成的是利用APICloud云编译的功能编译源代码生成独立的手机端的安装文件（Android的，IOS的），本实验编译选择的是Android的环境，在云编译选项中还可以选择IOS环境生成IOS手机下同下的安装文件。

需要注意：如果想生成测试版的安装程序可以不添加证书，如果生成正式版的安装文件则必须提供相关平台的开发证书，关于证书的了解参考：

## 1.4 实验四 其他开发环境的搭建

### 1.4.1 实验目的

1. 了解其他APICloud应用程序的开发环境搭建。

### 1.4.2 准备工作

无

### 1.4.3 实验步骤

1. WebStorm工具的使用。参考：<http://docs.apicloud.com/Dev-Tools/webStorm-apicloud-plugin>。

### 1.4.4 实验结论

了解APICloud Studio开发环境的灵活性，适合于不同的开发背景的人接触APICloud开发。

# 第2章 APICloud前端框架

## 2.1 实验一

### 2.1.1 实验目的

### 2.1.2 准备工作

无

### 2.1.3 实验步骤

### 2.1.4 实验结论

# 第3章 APICloud 布局和导航设计

## 3.1 实验一

### 3.1.1 实验目的

### 3.1.2 准备工作

无

### 3.1.3 实验步骤

### 3.1.4 实验结论

# 第4章 APICloud SuperWebView及模块开发

## 4.1 实验一 动态编译独立的SuperWebView SDK

### 4.1.1 实验目的

1. 完成SuperWebView动态SDK编译和生成，熟练动态编译SDK的过程。

### 4.1.2 准备工作

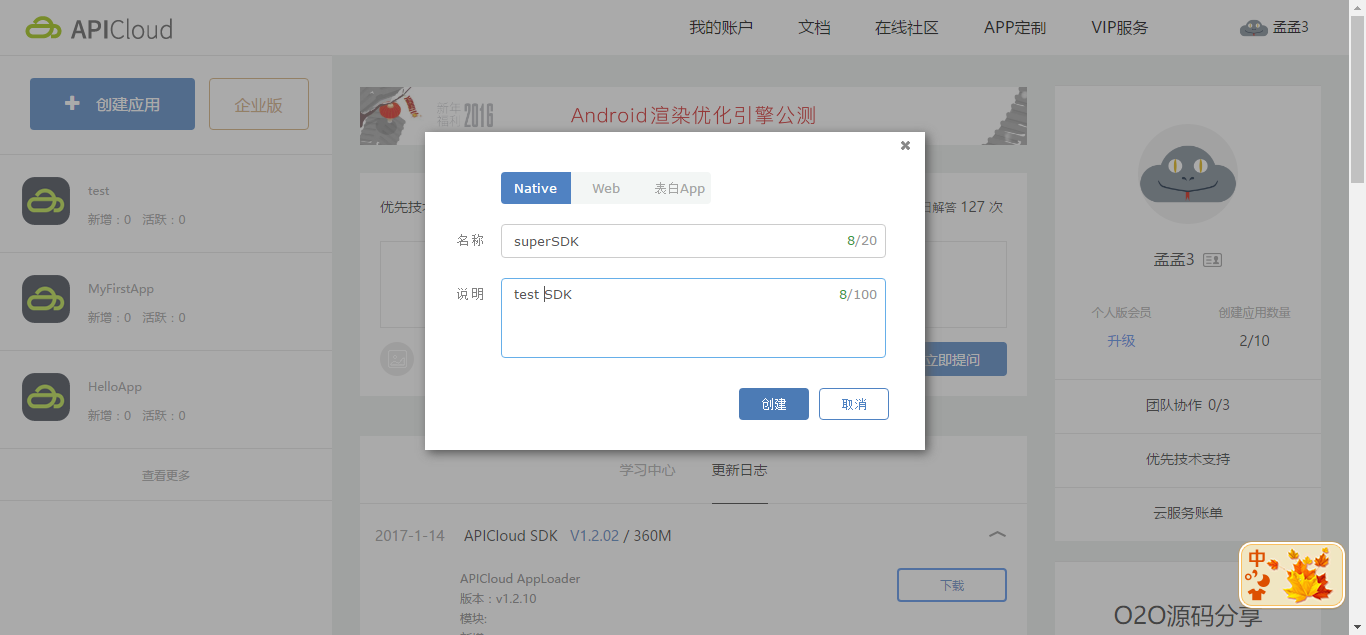
无

### 4.1.3 实验步骤

1. 创建APICloud工程，在APICloud平台上创建工程，比如工程名称为superSDK，如下图所示：



*图4.1.1 创建应用按钮*



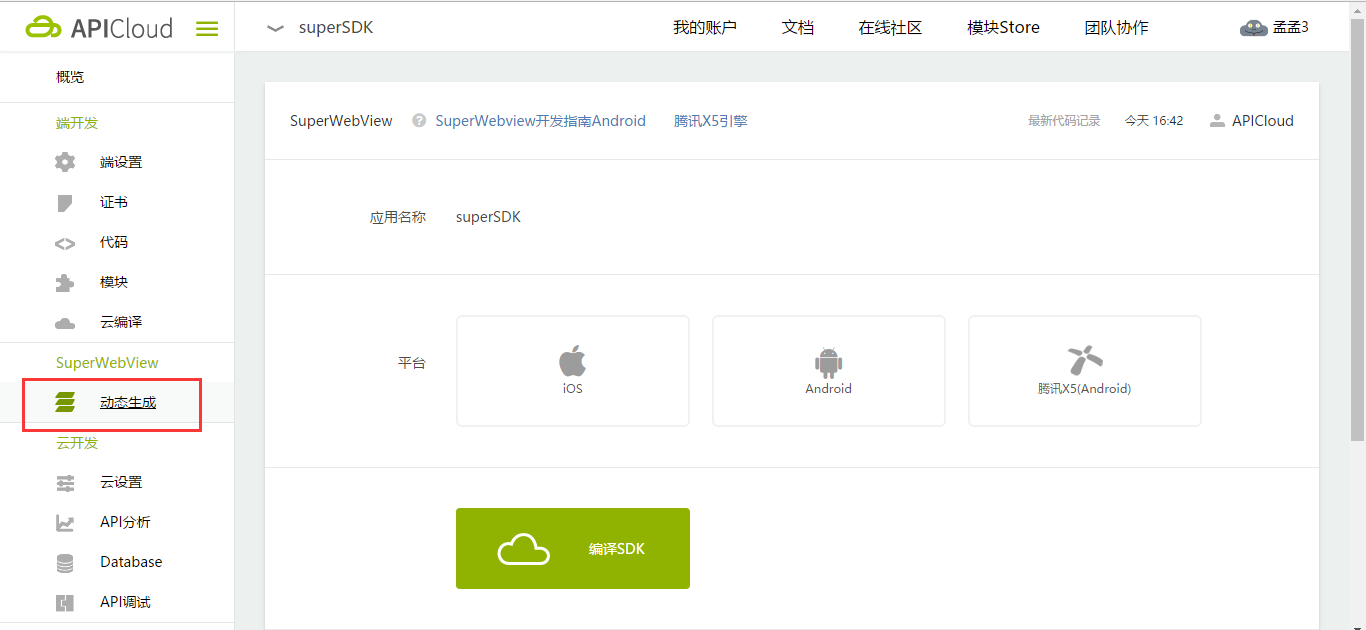
*图4.1.2 创建应用名称*

1. 进入新创建的应用的模块页面，如下图所示：选择Android项目中将要用到的模块，如果不需要，则略过此步骤。



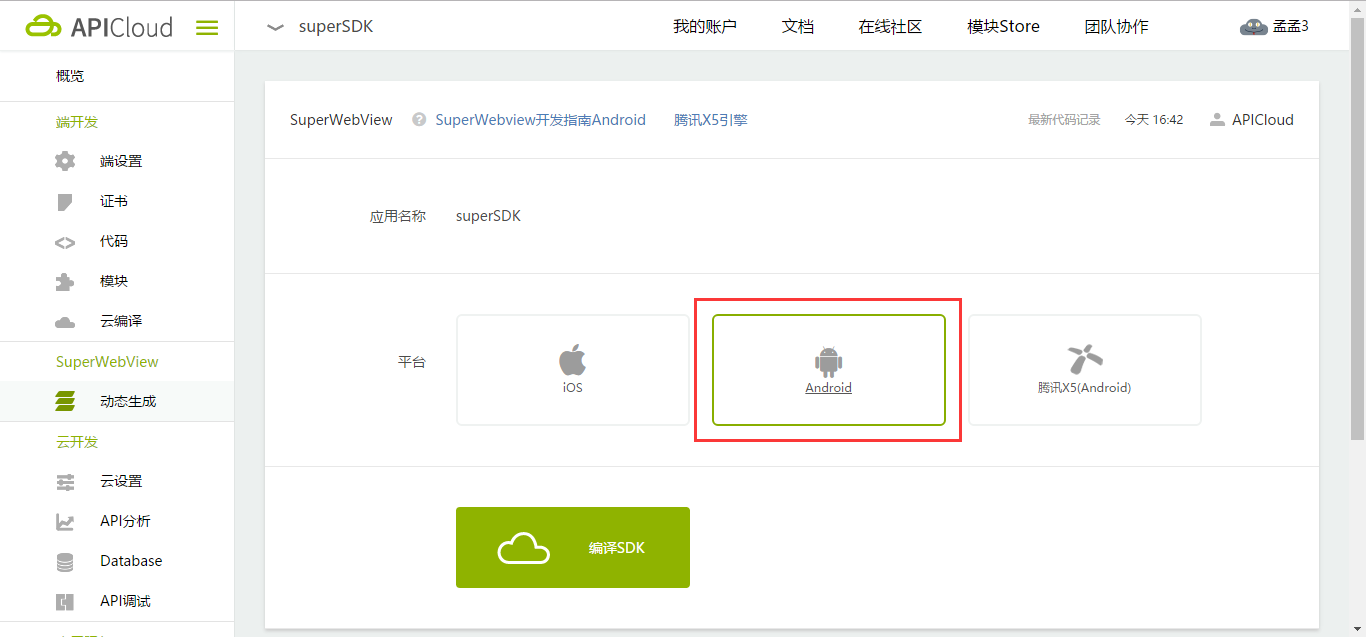
*图4.1.3 添加模块*

1. 选择左侧的“动态生成”链接，如下图所示：



*图4.1.4 动态生成界面*

1. 在动态生成界面，选择适合自己的平台，本实验是基于Android的开发平台，所以直接选择Android即可，如下图所示：



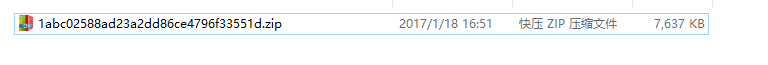
*图4.1.5 选择编译的平台*

1. 点击编译SDK绿色按钮，生成SDK成功后，如下图所示：



*图4.1.6 编译成功后界面*

下载SDK即可，如下图所示：



*图4.1.7 SDK下载成功*

### 4.1.4 实验结论

成功完成SuperWebView SDK动态编译的过程，在实际的应用中，原生应用中需要使用APICloud平台的那部分功能动态的SDK中就需要包含哪些功能对应的模块，这也是动态SuperWebView编译的优势和意义所在。

## 4.2 实验二 在原生的Android应用中引用SDK

### 4.2.1 实验目的

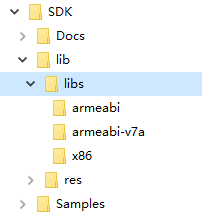
1. 掌握在原生的Android应用开发中使用APICloud平台的SuperWebView SDK。

### 4.2.2 准备工作

已经动态编译好待用的SuperWebView SDK包。

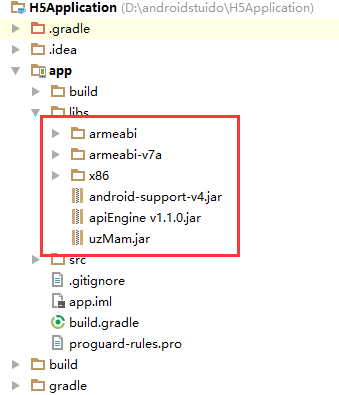
### 4.2.3 实验步骤

1. 解压实验一中动态编译生成的SuperWebView压缩包，得到SDK包，如下图所示：



*图4.2.1 解压后的SDK*

1. 复制lib/libs目录下所有的文件、文件夹到Android工程的libs目录，如下图所示：



*图4.2.2 Android工程引入SuperWebView SDK*

1. 将lib/res目录下的所有资源文件拷贝到Android工程对应的res同名目录中，注意values类型资源的合并。
2. lib/AndroidManifest.xml文件中的所有permission、activity、provider、service、receiver等全部拷贝到Android工程对应的AndroidManifest.xml中。
3. 如果有lib/assets目录，则将lib/assets目录下的所有资源文件拷贝到Android工程对应的assets目录中。
4. 开始使用SuperWebView，自定义Android工程的Application类，如下图所示：



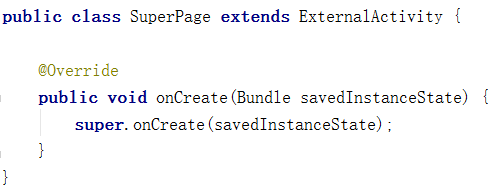
*图4.2.3 自定义Application*

并在AndroidManifest.xml文件中配置MyApplication，如下图所示：



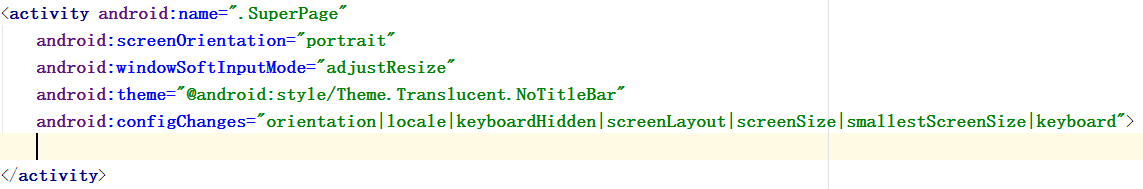
*图4.2.4 自定义Application配置*

1. 新建Activity类，继承自ExternalActivity（ExternalActivity的帮助说明请参考第五章重要API说明中的介绍，或者阅读API文档中的详细介绍），如下图所示：



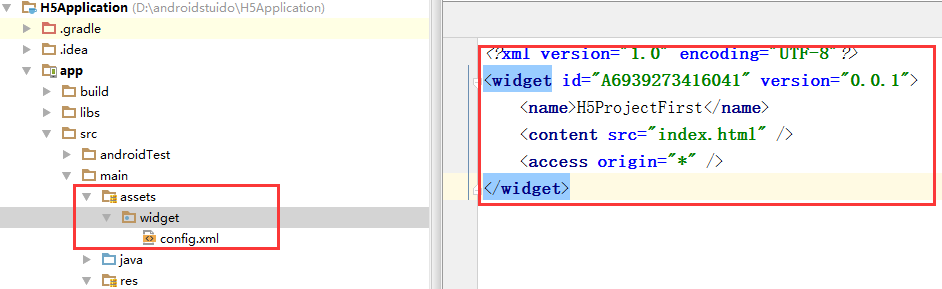
*图4.2.5 新建Activity*

同时，在AndroidManifest中配置SuperPage，如下图所示：



*图4.2.6 新建Activity配置*

1. H5代码的编写和配置，在assets目录下新建名为widget的目录（注意：Android Studio工具中默认创建的Android工程没有assets目录，需要手动创建，直接在src\main目录下新建assets目录，如图4.2.7所示），并在该目录下新建名为config.xml文件，config.xml文件要求编码为无BOM头UTF-8编码，如下图所示：



*图4.2.7*

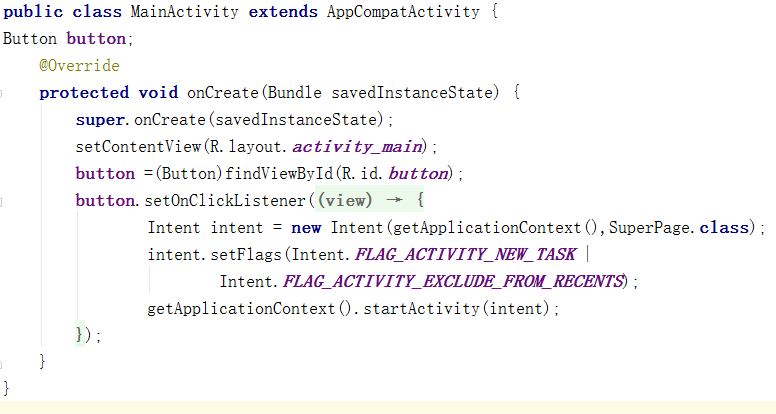
配置完成后，开始编写第一张Html页面，例如index.html，并将其在config.xml中配置为content字段，即声明该Html为SuperWebview的默认入口Html页。

关于widget包的结构请参考<http://docs.apicloud.com/Dev-Guide/widget-package-structure-manual>。

关于Html代码的编写，也可通过下载[APICloud开发工具](http://www.apicloud.com/dev)进行项目的创建，编码，调试，开发完成后直接将整个项目的代码拷贝到Android工程中的assets/widget目录下即可。

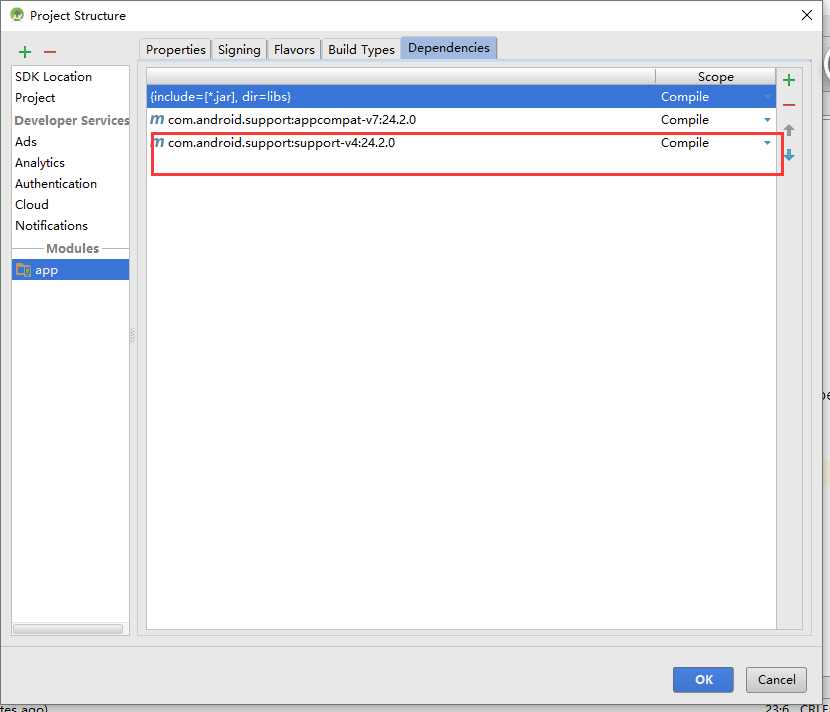
在此，将最初的APICloud项目MyFirstApp项目编写的html代码放入到assets/widget目录下。

1. 启动SuperWebView，设置首页中的按钮的点击事件，启动SuperWebView，如下图所示：



*图4.2.8*

1. 如果运行过程中出现Android的v4包与v7包冲突，则直接删除掉SuperWebView中libs中的v4包，在Android项目中动态依赖v4包，如下图所示：



*图4.2.9*

由于工程引入了第三方的jar包和so库，为了能在编译生成的apk文件中包含so库文件需要对Android工程的编译做设置，在build.grald文件中加入如下配置：



*图4.2.10*

1. 运行Android项目，效果如下图所示：

*图4.2.11*

### 4.2.4 实验结论

成功地在Android原生应用中使用SuperWebView SDK开发H5的页面，取代原生应用中的WebView组件。

# 第5章 APICloud云服务

## 5.1 实验一

### 5.1.1 实验目的

### 5.1.2 准备工作

无

### 5.1.3 实验步骤