# H5混合开发

android的原生控件与WebView进行混合开发。其实使用起来也是很简单，就一个WebView，外加本地的java代码与H5中的JavaScript进行交互。

涉及的内容包括：（1）WebView加载H5页面；

1. Android调用H5中的方法；
2. H5调用Android中的方法；
3. WebView确定速度测试、优化；（复用，而不是每次都重新启动。）
4. H5分模块更新逻辑；（较少更新的代价、流量少，部分更新。）
5. H5缓存使用的问题。（分为网页数据缓存和H5缓存，涉及缓存，必然就是要设置必要的网络、网络状态以及存储等权限，注意几种不同的缓存模式。）
6. WebView的注意内存泄漏问题。（2种方法：a.不在xml布局中添加webView控件，而是通过动态加载，在onCreate()方法体中加载，之后在onDestroy()中注意销毁。 b.另起一个进程加载webview，页面销毁后干掉这个进程。使用起来就比较麻烦了，涉及进程间通信AIDL，另起进程会导致CPU占用，但是优点是回收处理的比较好。）
7. Native App：本地应用程序（原生App）；Web App：网页应用程序（移动web）；Hybrid App：混合应用程序（混合App）。hybrid杂种的、混合的。
8. React Native App混合开发。

# 2、webview如何加载H5页面

我们都知道在Android中是通过webView来加载html页面的，根据HTML文件所在的位置不同写法也不同：

//加载assets文件夹下的test.html页面

mWebView.loadUrl("file:///android\_asset/test.html");

//加载网页

mWebView.loadUrl("http://www.baidu.com");

1.载入远程页面

（1）在AndroidManifest.xml文件中添加权限

<uses-permission android:name="android.permission.INTERNET"/>

（2）布局文件webview.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<WebView

android:id="@+id/wv\_webview"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" />

</LinearLayout>

（3）MainActivity.java

public class MainActivity extends AppCompatActivity {

WebView webView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.webview);

webView = (WebView)findViewById(R.id.wv\_webview);

loadWeb();

}

public void loadWeb(){

//String url = "https://www.baidu.com/";

String url = "file:///android\_asset/index.html";

//此方法可以在webview中打开链接而不会跳转到外部浏览器

webView.setWebViewClient(new WebViewClient());

webView.loadUrl(url);

}

//重载onKeyDown的函数，使其在页面内回退,而不是直接退出程序

public boolean onKeyDown(int keyCode, KeyEvent event) {

if(keyCode == KeyEvent.KEYCODE\_BACK && webView.canGoBack()){

webView.goBack();

return true;

}

return super.onKeyDown(keyCode, event);

}

}

2.载入本地页面

（1）新建assets目录用来存放html页面

　　　　 右键app->new->folder->assetsfolder

（2）html页面放入assets目录

　　（3）修改url

　　 webview.loadUrl(file:///android\_asset/index.html);

# 3、Android如何调用H5中的方法

（1）添加webview对调用js方法的支持

　　//支持html中javascript解析，不管是不是js和android交互，只要网页中含有js都要

　　webView.getSettings().setJavaScriptEnabled(true);

　　//支持弹窗，也就是支持html网页弹框

　　webView.setWebChromeClient(new WebChromeClient(){

　　public boolean onJsAlert(WebView view,String url,String message,JsResult result ){

　　　　　　return super.onJsAlert(view,url,message,result);

　　　　　　}

　　　　　　});

（2）调用h5无参无返回值的方法（直接调用）

　　　　　　h5方法：

　　　　　　　function show(){

　　　　　　　　document.getElementById("temp").innerHTML = "Hello world";

　　　　　　　}

　　　　　调用：

　　　　　 webView.loadUrl("javascript:show()");

（3）调用h5有参无返回值的方法

　　当调用H5带参数的方法时，势必要传入一个字符串，当传入固定字符串时，用单引号括起来即可；当传入变量名时，需要用转义符。

　　　　　　　h5方法：

　　　　　　　function alertMsg(message){

　　　　　　alert(message);

　　　　　　　}

　　　　　　　调用：

　　　　　　　webView.loadUrl("javascript:alertMsg('哈哈')");

　　　　　　　String content = "9880";

　　　　　　　webView.loadUrl("javascript:alertMsg(\""+content+"\")");

（4）调用h5有返回值方法

　　　　　　　function sum(i,j){

　　　　　　 return i+j;

　　　　　　　}

　　　　　　调用：

　　　　　　webView.evaluateJavascript("sum(1,2)",new ValueCallback() {

　　　　　　@Override

　　　　　　public void onReceiveValue(Object value) {

　　　　　　Toast.makeText(MainActivity.this,"js返回结果为="+value,Toast.LENGTH\_LONG).show();

　　　　　　}

　　　　　　});

完整代码：

index.html

<!DOCTYPE html>

<html>

<head>

<title>测试</title>

</head>

<body onLoad="init();">

<div id="temp">h5页面</div>

</body>

<script type="text/javascript">

//无参无返回值的方法

function show(){

document.getElementById("temp").innerHTML = "Hello world";

}

//有返回值的方法

function sum(i,j){

return i+j;

}

//有参无返回值的方法

function alertMsg(message){

alert(message);

}

</script>

</html>

webview.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<Button

android:id="@+id/btn"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Button"

android:layout\_marginBottom="12dp"

android:layout\_marginTop="8dp"/>

<WebView

android:id="@+id/wv\_webview"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" >

</WebView>

</LinearLayout>

MainActivity.java

public class MainActivity extends AppCompatActivity {

WebView webView;

Button btn;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.webview);

btn = (Button)findViewById(R.id.btn);

btn.setOnClickListener(new View.OnClickListener(){

public void onClick(View v){

//调用H5无参无返回值方法

webView.loadUrl("javascript:show()");

//调用H5有参方法

webView.loadUrl("javascript:alertMsg('哈哈')");

String content = "9880";

webView.loadUrl("javascript:alertMsg(\""+content+"\")");

//调用H5有返回值方法

webView.evaluateJavascript("sum(1,2)",new ValueCallback() {

@Override

public void onReceiveValue(Object value) {

Toast.makeText(MainActivity.this,"js返回结果为="+value,Toast.LENGTH\_LONG).show();

}

});

}

});

webView = (WebView)findViewById(R.id.wv\_webview);

loadWeb();

}

@SuppressLint("JavascriptInterface")

public void loadWeb(){

String url = "file:///android\_asset/index.html";

//此方法可以在webview中打开链接而不会跳转到外部浏览器

webView.setWebViewClient(new WebViewClient());

webView.loadUrl(url);

//支持弹窗，也就是支持html网页弹框

webView.setWebChromeClient(new WebChromeClient(){

public boolean onJsAlert(WebView view,String url,String message,JsResult result ){

return super.onJsAlert(view,url,message,result);

}

});

//支持html中javascript解析，不管是不是js和android交互，只要网页中含有js都要

webView.getSettings().setJavaScriptEnabled(true);

}

public boolean onKeyDown(int keyCode, KeyEvent event) {

//重写onKeyDown，当浏览网页，WebView可以后退时执行后退操作。

if(keyCode == KeyEvent.KEYCODE\_BACK && webView.canGoBack()){

webView.goBack();

return true;

}

return super.onKeyDown(keyCode,event);

}

}

# 4、H5如何调用Android中的方法

Android需要新建一个类，里边提供给H5操作的方法，并规定别名。

　　（1）新建一个类

　　　　public class JsInteration {

　　　　 @JavascriptInterface

　　　　public String back() {

　　　　return "hello world";

　　　　}

　　　　}

　　（2）webView添加js接口，并起别名

　　　　webView.addJavascriptInterface(new JsInteration(),"android");

　　（3）H5调用

　　　　window.android.back();

完整代码：

index.html

复制代码

<!DOCTYPE html>

<html>

<head>

<title>测试</title>

</head>

<body onLoad="init();">

<div id="temp">h5页面</div>

<button id="btn" onclick="getMsg()">按钮</button>

</body>

<script type="text/javascript">

function getMsg(){

var result=window.android.back();

document.getElementById("temp").innerHTML=result;

}

</script>

</html>

复制代码

MainActivity.java

复制代码

public class MainActivity extends AppCompatActivity {

WebView webView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.webview);

webView = (WebView)findViewById(R.id.wv\_webview);

loadWeb();

}

@SuppressLint("JavascriptInterface")

public void loadWeb(){

String url = "file:///android\_asset/dj\_index.html";

//此方法可以在webview中打开链接而不会跳转到外部浏览器

webView.setWebViewClient(new WebViewClient());

webView.loadUrl(url);

/\*

打开js接口，参数1为本地类名；参数2为别名

\*/

webView.addJavascriptInterface(new JsInteration(),"android");

}

public class JsInteration {

@JavascriptInterface//一定要写，不然h5调不到这个方法

public String back() {

return "hello world";

}

}

public boolean onKeyDown(int keyCode, KeyEvent event) {

//重写onKeyDown，当浏览网页，WebView可以后退时执行后退操作。

if(keyCode == KeyEvent.KEYCODE\_BACK && webView.canGoBack()){

webView.goBack();

return true;

}

return super.onKeyDown(keyCode,event);

}

}

# 关于内存泄漏

（1）不在xml布局中添加webview标签，采用在代码中new出来的方式。

再接着当页面销毁时，调用一些各种各样webview提供的方法，回收资源,比如：

ViewParent parent = mWebView.getParent();

if (parent != null) {

((ViewGroup) parent).removeView(mWebView);

}

mWebView.stopLoading();

mWebView.getSettings().setJavaScriptEnabled(false);

mWebView.clearHistory();

mWebView.clearView();

mWebView.removeAllViews();

..........................

mWebView.destroy()；

mWebView=null；

问题：这种方式有一定的作用。但是，退出页面后，并不能完全降到之前的内存水平。

有待改进。

1. 因此采取了另一个方案，另起一个进程加载webview，页面销毁后干掉这个进程。来看一下此图，ServcieLoginAcvitiy销魂后，内存几乎回到跟进入前差不多的水平值。

当然此方案拥有不可忽视的问题。什么问题呢？webview有业务逻辑需要交互，返回给主进程处理，于是就牵扯到进程间通信的问题。也会面临进程不小心被干掉了（比如内存增长过快，GC需要回收部分内存，根据策略优先级回收的时候，可能就会干掉这个进程），导致主进程无法接受到数据或者说其他的一些交流，出现严重的体验问题。

如果这个webview只是单纯地加载浏览界面，没有复杂的其他逻辑（比如与Js的交互），倒是不妨可以考虑。另起一个进程，要占用CPU时间，主进程需要等待接受信息，那么势必会慢一些。

总而言之，它的优势、劣势非常明显。需要考虑具体的业务场景和环境。

开启一个进程的方式：销毁内存的更干净。但是涉及到进程间通信，麻烦了；还有，还需要考虑进程被回收的可能性。