Webview使用介绍

**1.简介**

WebView是一个基于webkit引擎、展现web页面的控件。   
Android的Webview在低版本和高版本采用了不同的webkit版本内核，4.4后直接使用了Chrome。

**2.作用**

* 显示和渲染Web页面
* 直接使用html文件（网络上或本地assets中）作布局
* 可和JavaScript交互调用

WebView控件功能强大，除了具有一般View的属性和设置外，还可以对url请求、页面加载、渲染、页面交互进行强大的处理。

**3.使用介绍**

一般来说Webview可单独使用，可联合其子类一起使用，下面会介绍几种:

* Webview自身的常见方法；
* Webview的最常用的子类   
  （WebSettings类、WebViewClient类、WebChromeClient类)
* Android和Js的交互

**3.1 Webview常用方法**

**3.1.1 加载URL**

可以根据资源分为三种:

1. 加载一个网页：   
webView.loadUrl(“<http://www.baidu.com/>“);

2：加载apk包中的html页面   
webView.loadUrl(“file:///android\_asset/test.html”);

//方式3：加载手机本地的html页面   
webView.loadUrl(“content://com.android.htmlfileprovider/sdcard/test.html”);

// 方式4： 加载 HTML 页面的一小段内容   
WebView.loadData(String data, String mimeType, String encoding)   
// 参数说明：   
// 参数1：需要截取展示的内容   
// 内容里不能出现 ’#’, ‘%’, ‘\’ , ‘?’ 这四个字符，若出现了需用 %23, %25, %27, %3f 对应来替代，否则会出现异常   
// 参数2：展示内容的类型   
// 参数3：字节码

**3.1.2 WebView的状态**

(1)激活WebView为活跃状态，能正常执行网页的响应   
webView.onResume() ；

(2)当页面被失去焦点被切换到后台不可见状态，需要执行onPause   
//通过onPause动作通知内核暂停所有的动作，比如DOM的解析、plugin的执行、JavaScript执行。   
webView.onPause()；

(3)当应用程序(存在webview)被切换到后台时，这个方法不仅仅针对当前的webview而是全局的全应用程序的webview   
//它会暂停所有webview的layout，parsing，javascripttimer。降低CPU功耗。   
webView.pauseTimers()   
(4)恢复pauseTimers状态   
webView.resumeTimers()；

(4)销毁Webview   
//在关闭了Activity时，如果Webview的音乐或视频，还在播放。就必须销毁Webview   
//但是注意：webview调用destory时,webview仍绑定在Activity上   
//这是由于自定义webview构建时传入了该Activity的context对象   
//因此需要先从父容器中移除webview,然后再销毁webview:   
rootLayout.removeView(webView);   
webView.destroy();

**3.1.2 关于前进 / 后退网页**

//是否可以后退   
Webview.canGoBack()   
//后退网页   
Webview.goBack()

//是否可以前进   
Webview.canGoForward()   
//前进网页   
Webview.goForward()

//以当前的index为起始点前进或者后退到历史记录中指定的steps   
//如果steps为负数则为后退，正数则为前进   
Webview.goBackOrForward(intsteps)

**常见用法：Back键控制网页后退**

* 问题：在不做任何处理前提下 ，浏览网页时点击系统的“Back”键,整个 Browser 会调用 finish()而结束自身
* 目标：点击返回后，是网页回退而不是推出浏览器
* 解决方案：在当前Activity中处理并消费掉该 Back 事件   
  public boolean onKeyDown(int keyCode, KeyEvent event) {   
  if ((keyCode == KEYCODE\_BACK) && mWebView.canGoBack()) {   
  mWebView.goBack();   
  return true;   
  }   
  return super.onKeyDown(keyCode, event);   
  }

**3.1.3 清除缓存数据**

(1)清除网页访问留下的缓存   
//由于内核缓存是全局的因此这个方法不仅仅针对webview而是针对整个应用程序.   
Webview.clearCache(true);

(2)清除当前webview访问的历史记录   
//只会webview访问历史记录里的所有记录除了当前访问记录   
Webview.clearHistory()；

(3)这个api仅仅清除自动完成填充的表单数据，并不会清除WebView存储到本地的数据   
Webview.clearFormData()；

总结：WebView处理的主要是View，1.指定加载的HTML（loadurl()），2.设置View的状态(onResume, onPause, pauseTimers, resumeTimers 后两个是全局的，当activity不可见时使用，可以降低电量和资源消耗) 3.页面(View)的前进(goForward)和后退(goBack),前进或者后退 4.清理缓存

**3.2 常用类**

**3.2.1 WebSettings类**

\* 作用：对WebView进行配置和管理

\* 配置步骤 & 常见方法：

**配置步骤1：添加访问网络权限（AndroidManifest.xml）**   
这是前提！这是前提！这是前提！   
  
**配置步骤2：生成一个WebView组件（有两种方式）**

//方式1：直接在在Activity中生成   
WebView webView = new WebView(this)

//方法2：在Activity的layout文件里添加webview控件：   
WebView webview = (WebView) findViewById(R.id.webView1);   
**配置步骤3：进行配置-利用WebSettings子类（常见方法）**   
//声明WebSettings子类   
WebSettings webSettings = webView.getSettings();

1. 如果访问的页面中要与Javascript交互，则webview必须设置支持Javascript   
   webSettings.setJavaScriptEnabled(true);

注意：若加载的 html 里有JS 在执行动画等操作，会造成资源浪费（CPU、电量）   
// 在 onStop 和 onResume 里分别把 setJavaScriptEnabled() 给设置成 false 和 true 即可

(2)支持插件   
webSettings.setPluginsEnabled(true);

(3)设置自适应屏幕，两者合用   
webSettings.setUseWideViewPort(true); //将图片调整到适合webview的大小   
webSettings.setLoadWithOverviewMode(true); // 缩放至屏幕的大小

(4)缩放操作   
webSettings.setSupportZoom(true); //支持缩放，默认为true。是下面那个的前提。   
webSettings.setBuiltInZoomControls(true); //设置内置的缩放控件。若为false，则该WebView不可缩放   
webSettings.setDisplayZoomControls(false); //隐藏原生的缩放控件

(5)其他细节操作   
webSettings.setCacheMode(WebSettings.LOAD\_CACHE\_ELSE\_NETWORK); //关闭webview中缓存   
webSettings.setAllowFileAccess(true); //设置可以访问文件   
webSettings.setJavaScriptCanOpenWindowsAutomatically(true); //支持通过JS打开新窗口   
webSettings.setLoadsImagesAutomatically(true); //支持自动加载图片   
webSettings.setDefaultTextEncodingName(“utf-8”);//设置编码格式

**常见用法：设置WebView缓存**

\* 当加载 html 页面时，WebView会在/data/data/包名目录下生成 database 与 cache 两个文件夹

\* 请求的 URL记录保存在 WebViewCache.db，而 URL的内容是保存在 WebViewCache 文件夹下

\* 是否启用缓存：

(6)优先使用缓存:   
WebView.getSettings().setCacheMode(WebSettings.LOAD\_CACHE\_ELSE\_NETWORK);   
(6.1)缓存模式如下：   
//LOAD\_CACHE\_ONLY: 不使用网络，只读取本地缓存数据   
//LOAD\_DEFAULT: （默认）根据cache-control决定是否从网络上取数据。   
//LOAD\_NO\_CACHE: 不使用缓存，只从网络获取数据.   
//LOAD\_CACHE\_ELSE\_NETWORK，只要本地有，无论是否过期，或者no-cache，都使用缓存中的数据。

//不使用缓存:   
WebView.getSettings().setCacheMode(WebSettings.LOAD\_NO\_CACHE);   
\* 结合使用（离线加载）

if (NetStatusUtil.isConnected(getApplicationContext())) {   
webSettings.setCacheMode(WebSettings.LOAD\_DEFAULT);//根据cache-control决定是否从网络上取数据。   
} else {   
webSettings.setCacheMode(WebSettings.LOAD\_CACHE\_ELSE\_NETWORK);//没网，则从本地获取，即离线加载   
}

webSettings.setDomStorageEnabled(true); // 开启 DOM storage API 功能   
webSettings.setDatabaseEnabled(true); //开启 database storage API 功能   
webSettings.setAppCacheEnabled(true);//开启 Application Caches 功能

String cacheDirPath = getFilesDir().getAbsolutePath() + APP\_CACAHE\_DIRNAME;   
webSettings.setAppCachePath(cacheDirPath); //设置 Application Caches 缓存目录   
注意： 每个 Application 只调用一次 WebSettings.setAppCachePath()，WebSettings.setAppCacheMaxSize()

总结:WebSetting的作用：1.设置是否支持js（setJavaScriptEnabled()有的就是会很消耗电量和CPU）2.设置屏幕自适应setLoadWithOverviewMode(true)；屏幕缩放setSupportZoom(true) 3.缓存（setCacheMode()）：是否缓存;优先使用本地缓存还是网络；设置缓存目录；4.是否开启Dom和Database 存储功能（每个 Application 只调用一次 WebSettings.setAppCachePath()，WebSettings.setAppCacheMaxSize()）

**3.2.2 WebViewClient类**

\* 作用：处理各种通知 & 请求事件

\* 常见方法：

**常见方法1：shouldOverrideUrlLoading()**

\* 作用：打开网页时不调用系统浏览器， 而是在本WebView中显示；在网页上的所有加载都经过这个方法,这个函数我们可以做很多操作。

**//步骤1. 定义Webview组件   
Webview webview = (WebView) findViewById(R.id.webView1);**

**//步骤2. 选择加载方式   
//方式1. 加载一个网页：   
webView.loadUrl(“**[**http://www.google.com/**](http://www.google.com/)**“);**

**//方式2：加载apk包中的html页面   
webView.loadUrl(“file:///android\_asset/test.html”);**

**//方式3：加载手机本地的html页面   
webView.loadUrl(“content://com.android.htmlfileprovider/sdcard/test.html”);**

**//步骤3. 复写shouldOverrideUrlLoading()方法，使得打开网页时不调用系统浏览器， 而是在本WebView中显示   
webView.setWebViewClient(new WebViewClient(){   
@Override   
public boolean shouldOverrideUrlLoading(WebView view, String url) {   
view.loadUrl(url);   
return true;   
}   
});**

**常见方法2：onPageStarted()**

\* 作用：开始载入页面调用的，我们可以设定一个loading的页面，告诉用户程序在等待网络响应。

webView.setWebViewClient(new WebViewClient(){   
@Override   
public void onPageStarted(WebView view, String url, Bitmap favicon) {   
//设定加载开始的操作   
}   
});

**常见方法3：onPageFinished()**

\* 作用：在页面加载结束时调用。我们可以关闭loading 条，切换程序动作。

webView.setWebViewClient(new WebViewClient(){   
@Override   
public void onPageFinished(WebView view, String url) {   
//设定加载结束的操作   
}   
});

**常见方法4：onLoadResource()**

\* 作用：在加载页面资源时会调用，每一个资源（比如图片）的加载都会调用一次。

webView.setWebViewClient(new WebViewClient(){   
@Override   
public boolean onLoadResource(WebView view, String url) {   
//设定加载资源的操作   
}   
});

**常见方法5：onReceivedError（）**

\* 作用：加载页面的服务器出现错误时（如404）调用。

App里面使用webview控件的时候遇到了诸如404这类的错误的时候，若也显示浏览器里面的那种错误提示页面就显得很丑陋了，那么这个时候我们的app就需要加载一个本地的错误提示页面，即webview如何加载一个本地的页面

//步骤1：写一个html文件（error\_handle.html），用于出错时展示给用户看的提示页面   
//步骤2：将该html文件放置到代码根目录的assets文件夹下

//步骤3：复写WebViewClient的onRecievedError方法   
//该方法传回了错误码，根据错误类型可以进行不同的错误分类处理   
webView.setWebViewClient(new WebViewClient(){   
@Override   
public void onReceivedError(WebView view, int errorCode, String description, String failingUrl){   
switch(errorCode)   
{   
case HttpStatus.SC\_NOT\_FOUND:   
view.loadUrl(“file:///android\_assets/error\_handle.html”);   
break;   
}   
}   
});   
**常见方法6：onReceivedSslError()**

\* 作用：处理https请求

webView默认是不处理https请求的，页面显示空白，需要进行如下设置：

webView.setWebViewClient(new WebViewClient() {   
@Override   
public void onReceivedSslError(WebView view, SslErrorHandler handler, SslError error) {   
handler.proceed(); //表示等待证书响应   
// handler.cancel(); //表示挂起连接，为默认方式   
// handler.handleMessage(null); //可做其他处理   
}   
});

**3.2.3 WebChromeClient类**

\* 作用：辅助 WebView 处理 Javascript 的对话框,网站图标,网站标题等等。

\* 常见使用：

**常见方法1： onProgressChanged（）**

\* 作用：获得网页的加载进度并显示

webview.setWebChromeClient(new WebChromeClient(){

@Override   
public void onProgressChanged(WebView view, int newProgress) {   
if (newProgress < 100) {   
String progress = newProgress + “%”;   
progress.setText(progress);   
} else {   
}   
});

**常见方法2： onReceivedTitle（）**

\* 作用：获取Web页中的标题

每个网页的页面都有一个标题，比如www.baidu.com这个页面的标题即“百度一下，你就知道”，那么如何知道当前webview正在加载的页面的title并进行设置呢？

webview.setWebChromeClient(new WebChromeClient(){

@Override   
public void onReceivedTitle(WebView view, String title) {   
titleview.setText(title)；   
}   
**常见方法3： onJsAlert（）**

\* 作用：支持javascript的警告框

一般情况下在 Android 中为 Toast，在文本里面加入\n就可以换行

webview.setWebChromeClient(new WebChromeClient() {

@Override   
public boolean onJsAlert(WebView view, String url, String message, final JsResult result) {   
new AlertDialog.Builder(MainActivity.this)   
.setTitle(“JsAlert”)   
.setMessage(message)   
.setPositiveButton(“OK”, new DialogInterface.OnClickListener() {   
@Override   
public void onClick(DialogInterface dialog, int which) {   
result.confirm();   
}   
})   
.setCancelable(false)   
.show();   
return true;   
}

**常见方法4： onJsConfirm（）**

\* 作用：支持javascript的确认框

webview.setWebChromeClient(new WebChromeClient() {

@Override   
public boolean onJsConfirm(WebView view, String url, String message, final JsResult result) {   
new AlertDialog.Builder(MainActivity.this)   
.setTitle(“JsConfirm”)   
.setMessage(message)   
.setPositiveButton(“OK”, new DialogInterface.OnClickListener() {   
@Override   
public void onClick(DialogInterface dialog, int which) {   
result.confirm();   
}   
})   
.setNegativeButton(“Cancel”, new DialogInterface.OnClickListener() {   
@Override   
public void onClick(DialogInterface dialog, int which) {   
result.cancel();   
}   
})   
.setCancelable(false)   
.show();   
// 返回布尔值：判断点击时确认还是取消   
// true表示点击了确认；false表示点击了取消；   
return true;   
}

**常见方法5： onJsPrompt（）**

\* 作用：支持javascript输入框

点击确认返回输入框中的值，点击取消返回 null。

webview.setWebChromeClient(new WebChromeClient() {   
@Override   
public boolean onJsPrompt(WebView view, String url, String message, String defaultValue, final JsPromptResult result) {   
final EditText et = new EditText(MainActivity.this);   
et.setText(defaultValue);   
new AlertDialog.Builder(MainActivity.this)   
.setTitle(message)   
.setView(et)   
.setPositiveButton(“OK”, new DialogInterface.OnClickListener() {   
@Override   
public void onClick(DialogInterface dialog, int which) {   
result.confirm(et.getText().toString());   
}   
})   
.setNegativeButton(“Cancel”, new DialogInterface.OnClickListener() {   
@Override   
public void onClick(DialogInterface dialog, int which) {   
result.cancel();   
}   
})   
.setCancelable(false)   
.show();

return true;   
}

总结：WebViewClient：1.url加载之前的监听(shouldOverrideUrlLoading), 加载资源的监听（**onLoadResource**）例如，图片

2.页面开始加载（**onPageStarted**），结束加载（**onPageFinished**） ，加载进度(onProgressChanged)，http出错（**onReceivedError**）https加载出错(**onReceivedSslError**)

3.网页标题（**onReceivedTitle**），警告弹框(**onJsAlert**)，确定弹框（**onJsConfirm**），输入框（**onJsPrompt**）（这些可以替换Android弹出） 3.

## 3.3 WebView与JavaScript的交互

WebView的强大之处就是能和JavaScript进行交互调用。

### 3.3.1 java调用JavaScript

1.实例化webview，获得WebSettings。通过WebSettings设置可以和JavaScript交互。

myWebView = (WebView) findViewById(R.id.webview);

WebSettings webSettings = myWebView.getSettings();

// 使能使用JavaScript

webSettings.setJavaScriptEnabled(true);

2.设置webview的 WebViewClient，重写onPageFinished方法。设置当网页加载完后去调用网页中JavaScript。

myWebView.setWebViewClient(new WebViewClient() {

@Override

public void onPageFinished(WebView view, String url) {

//当page也加载完后

super.onPageFinished(view, url);

javaUseJavaScript(myWebView);

}

});

myWebView.loadUrl("file:///android\_asset/js\_java\_interaction.html");

3.实现第二步里自定义的javaUseJavaScript方法。

/\*\*

\* java调用js方法

\*

\* @param webView

\*/

private void javaUseJavaScript(WebView webView) {

String call = "javascript:sayHello()";

// call = "javascript:alertMessage(\"" + "content" + "\")";

//

// call = "javascript:toastMessage(\"" + "content" + "\")";

//

// call = "javascript:sumToJava(1,2)";

webView.loadUrl(call);

}

 3.完成了上面的步骤，就能调用JavaScript中sayHello()方法。sayHello方法在哪呢？？就在你myWebView.loadUrl("file:///android\_asset/js\_java\_interaction.html");的html文件中

<html>

<script type="text/javascript">

function sayHello() {

alert("Hello")

}

</script>

Java-Javascript Interaction In Android

</html>

总结：的从上面的代码可以看出，java调js的代码中，java可以给js传递参数，但无法获得返回值。如果必须要返回值，只能反过来通过js调用java。也就是我们下面说的：

### **3.3.2 JavaScript调用Java**

编写JavaScript代码去调用java中的方法。其实在**java调用JavaScript**的第三步的html中我们已经调用了。（绿色部分）

在java中编写代码，响应JavaScript的调用。

 2.1给webview添加JavaScript接口类

myWebView.addJavascriptInterface(new JsInteration(), " MyCount");

 2.2编写JavaScript的接口类  JsInteration

/\*\*

\* 实现js调用java的具体代码

\* @author cyq

\*

\*/

public class JsInteration {

@JavascriptInterface

public void toastMessage(String message) {

Toast.makeText(getApplicationContext(), message, Toast.LENGTH\_LONG).show();

}

@JavascriptInterface

public void onSumResult(int result) {

Log.i(TAG, "onSumResult result=" + result);

}

}

在html中

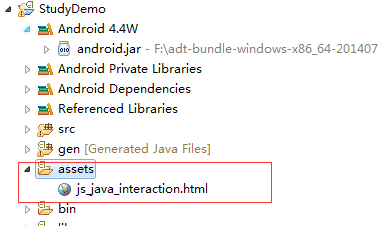
if(typeof MyCount.count === "function"){  
 MyCount.count(counts);  
}

总结：js调用java代码就是window.在addJavascriptInterface设置的第二个字符参数.在addJavascriptInterface这第一个参数类里的方法.(参数1，参数2.....).

注意：

                1要调用WebView的addJavascriptInterface方法。在JavaScript的接口类中的方法前要加上@JavascriptInterface注解。

                2html文件要放到Android项目的assets文件夹下：



                3 调用JavaScript要在加载完网页之后，就是WebViewClient的onPageFinished方法中。

                4 WebView的方法必须在主线程之中，至于为何就不知道了。原本我担心加载网页放在主线程中会导致ANR，其实不用担心这个问题，WebView内部应该做了处理了。

实际代码：<https://github.com/bigthing33/StudyDemo.git>

代码在WebViewActivity中。

### **3.3.4 Java调用Js代码（有返回值）**

前面说到java调用JavaScript只能传参数，无法得到返回值。如果需要返回值需要反过来JavaScript调用java代码，通过参数传递获得返回值。这在4.4以后有了改进。可以通过下面说的调用JavaScript的方法获得返回值。不过我个人认为还是用传递参数的方法比较好吧。版本兼容嘛。

在WebViewClient的onPageFinished方法中调用JavaScript

myWebView.setWebViewClient(new WebViewClient() {

@Override

public void onPageFinished(WebView view, String url) {

//当page也加载完后

super.onPageFinished(view, url);

testEvaluateJavascript( myWebView);

}

});

实现java代码

/\*\*

\* 调用JavaScript方法获得返回值

\* @param myWebView2

\*/

@SuppressLint("NewApi")

protected void testEvaluateJavascript(WebView myWebView2) {

myWebView.evaluateJavascript("getReturn(\""+"参数1"+"\")", new ValueCallback<String>() {

@Override

public void onReceiveValue(String value) {

Log.i(TAG, "onReceiveValue value=" + value);

}

});

}

在html中编写JavaScript代码（绿色部分）

<html>

<script type="text/javascript">

function getReturn(message) {

return 1;

}

</script>

Java-Javascript Interaction In Android

</html>

[复制代码](javascript:void(0);)

 运行代码，在Android的log中能够看到返回值1.

总结：

1.Android调用js方法

1. 在Android4.4.之前(无法拿到js方法返回值)

webView.loadUrl（"javascript:sayHello()"）; 注意：调用要在加载url完成之后（onPageFinished()）

webView.loadUrl要放在主线程中

@SuppressLint("NewApi")

protected void testEvaluateJavascript(WebView myWebView2) {

myWebView.evaluateJavascript("getReturn(\""+"参数1"+"\")", new ValueCallback<String>() {

@Override

public void onReceiveValue(String value) {

Log.i(TAG, "onReceiveValue value=" + value);

} }); }

2.Js调用Android代码

Js调用Android代码：” window. 字符参数.字符参数对应Android类的方法”

myWebView.addJavascriptInterface(new JsInteration(), "AndroidFunction");

function init()

{

var testVal = document.getElementById('mytextId').value;

AndroidFunction.showToast(testVal);

}

## 3.4 注意事项：如何避免WebView内存泄露？

### 3.4.1 不在xml中定义 Webview ，而是在需要的时候在Activity中创建，并且Context使用 getApplicationgContext()

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, ViewGroup.LayoutParams.MATCH\_PARENT);   
mWebView = new WebView(getApplicationContext());   
mWebView.setLayoutParams(params);   
mLayout.addView(mWebView);

### 3.4.2 在 Activity 销毁（ WebView ）的时候，先让 WebView 加载null内容，然后移除 WebView，再销毁 WebView，最后置空。

@Override   
protected void onDestroy() {   
if (mWebView != null) {   
mWebView.loadDataWithBaseURL(null, “”, “text/html”, “utf-8”, null);   
mWebView.clearHistory();

((ViewGroup) mWebView.getParent()).removeView(mWebView);   
mWebView.destroy();   
mWebView = null;   
}   
super.onDestroy();   
}