# **史上最全WebView使用，附送Html5Activity一份**

WebView在现在的项目中使用的频率应该还是非常高的。  
我个人总觉得HTML5是一种趋势。找了一些东西，在此总结。  
本篇最后有一个非常不错 的 Html5Activity 加载类，不想看的可以直接跳下载。

### WebSettings

WebSettings webSettings = mWebView .getSettings();

//支持获取手势焦点，输入用户名、密码或其他

webview.requestFocusFromTouch();

setJavaScriptEnabled(true); //支持js

setPluginsEnabled(true); //支持插件

webSettings.setRenderPriority(RenderPriority.HIGH); //提高渲染的优先级

设置自适应屏幕，两者合用

setUseWideViewPort(true); //将图片调整到适合webview的大小

setLoadWithOverviewMode(true); // 缩放至屏幕的大小

setSupportZoom(true); //支持缩放，默认为true。是下面那个的前提。

setBuiltInZoomControls(true); //设置内置的缩放控件。//若上面是false，则该WebView不可缩放，这个不管设置什么都不能缩放。

setDisplayZoomControls(false); //隐藏原生的缩放控件

setLayoutAlgorithm(LayoutAlgorithm.SINGLE\_COLUMN); //支持内容重新布局

supportMultipleWindows(); //多窗口

setCacheMode(WebSettings.LOAD\_CACHE\_ELSE\_NETWORK); //关闭webview中缓存

setAllowFileAccess(true); //设置可以访问文件

setNeedInitialFocus(true); //当webview调用requestFocus时为webview设置节点

setJavaScriptCanOpenWindowsAutomatically(true); //支持通过JS打开新窗口

setLoadsImagesAutomatically(true); //支持自动加载图片

setDefaultTextEncodingName("utf-8");//设置编码格式

### 关于缓存

##### 缓存模式

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。

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()

### 加载方式

**加载一个网页：**  
webView.loadUrl("[http://www.google.com/](http://www.google.com/" \t "_blank)");  
**加载apk包中的一个html页面**  
webView.loadUrl("file:///android\_asset/test.html");  
**加载手机本地的一个html页面的方法：**  
webView.loadUrl("content://com.android.htmlfileprovider/sdcard/test.html");

### WebViewClient

**WebViewClient就是帮助WebView处理各种通知、请求事件的。**  
打开网页时不调用系统浏览器， 而是在本WebView中显示：

mWebView.setWebViewClient(new WebViewClient(){

@Override

public boolean shouldOverrideUrlLoading(WebView view, String url) {

view.loadUrl(url);

return true;

}

});

WebViewClient方法

WebViewClient mWebViewClient = new WebViewClient()

{

shouldOverrideUrlLoading(WebView view, String url) 最常用的，比如上面的。

//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。

//比如获取url，查看url.contains(“add”)，进行添加操作

shouldOverrideKeyEvent(WebView view, KeyEvent event)

//重写此方法才能够处理在浏览器中的按键事件。

onPageStarted(WebView view, String url, Bitmap favicon)

//这个事件就是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。

onPageFinished(WebView view, String url)

//在页面加载结束时调用。同样道理，我们可以关闭loading 条，切换程序动作。

onLoadResource(WebView view, String url)

// 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。

shouldInterceptRequest(WebView view, String url)

// 拦截替换网络请求数据, API 11开始引入，API 21弃用

shouldInterceptRequest (WebView view, WebResourceRequest request)

// 拦截替换网络请求数据, 从API 21开始引入

onReceivedError(WebView view, int errorCode, String description, String failingUrl)

// (报告错误信息)

doUpdateVisitedHistory(WebView view, String url, boolean isReload)

//(更新历史记录)

onFormResubmission(WebView view, Message dontResend, Message resend)

//(应用程序重新请求网页数据)

onReceivedHttpAuthRequest(WebView view, HttpAuthHandler handler, String host,String realm)

//（获取返回信息授权请求）

onReceivedSslError(WebView view, SslErrorHandler handler, SslError error)

//重写此方法可以让webview处理https请求。

onScaleChanged(WebView view, float oldScale, float newScale)

// (WebView发生改变时调用)

onUnhandledKeyEvent(WebView view, KeyEvent event)

//（Key事件未被加载时调用）

}

**将上面定义的WebViewClient设置给WebView：**

webView.setWebViewClient(mWebViewClient);

### WebChromeClient

**WebChromeClient是辅助WebView处理Javascript的对话框，网站图标，网站title，加载进度等 :**  
方法中的代码都是由Android端自己处理。

WebChromeClient mWebChromeClient = new WebChromeClient() {

//获得网页的加载进度，显示在右上角的TextView控件中

@Override

public void onProgressChanged(WebView view, int newProgress) {

if (newProgress < 100) {

String progress = newProgress + "%";

} else {

}

}

//获取Web页中的title用来设置自己界面中的title

//当加载出错的时候，比如无网络，这时onReceiveTitle中获取的标题为 找不到该网页,

//因此建议当触发onReceiveError时，不要使用获取到的title

@Override

public void onReceivedTitle(WebView view, String title) {

MainActivity.this.setTitle(title);

}

@Override

public void onReceivedIcon(WebView view, Bitmap icon) {

//

}

@Override

public boolean onCreateWindow(WebView view, boolean isDialog, boolean isUserGesture, Message resultMsg) {

//

return true;

}

@Override

public void onCloseWindow(WebView window) {

}

//处理alert弹出框，html 弹框的一种方式

@Override

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

//

return true;

}

//处理confirm弹出框

@Override

public boolean onJsPrompt(WebView view, String url, String message, String defaultValue, JsPromptResult

result) {

//

return true;

}

//处理prompt弹出框

@Override

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

//

return true;

}

};

**同样，将上面定义的WebChromeClient设置给WebView：**

webView.setWebChromeClient(mWebChromeClient);

### 调用JS代码

WebSettings webSettings = mWebView .getSettings();

webSettings.setJavaScriptEnabled(true);

mWebView.addJavascriptInterface(new InsertObj(), "jsObj");

**上面这是前提！！！**  
然后实现上面的类，这个类提供了四个方法，注释的非常清楚。

class InsertObj extends Object {

//给html提供的方法，js中可以通过：var str = window.jsObj.HtmlcallJava(); 获取到

@JavascriptInterface

public String HtmlcallJava() {

return "Html call Java";

}

//给html提供的有参函数 ： window.jsObj.HtmlcallJava2("IT-homer blog");

@JavascriptInterface

public String HtmlcallJava2(final String param) {

return "Html call Java : " + param;

}

//Html给我们提供的函数

@JavascriptInterface

public void JavacallHtml() {

runOnUiThread(new Runnable() {

@Override

public void run() {

//这里是调用方法

mWebView.loadUrl("javascript: showFromHtml()");

Toast.makeText(Html5Activity.this, "clickBtn", Toast.LENGTH\_SHORT).show();

}

});

}

//Html给我们提供的有参函数

@JavascriptInterface

public void JavacallHtml2(final String param) {

runOnUiThread(new Runnable() {

@Override

public void run() {

mWebView.loadUrl("javascript: showFromHtml2('IT-homer blog')");

Toast.makeText(Html5Activity.this, "clickBtn2", Toast.LENGTH\_SHORT).show();

}

});

}

}

Android 调用js有个漏洞：  
[http://blog.csdn.net/leehong2005/article/details/11808557](http://blog.csdn.net/leehong2005/article/details/11808557" \t "_blank)

### Cookie 相关

之前同步 cookie 需要用到 CookieSyncManager 类，现在这个类已经被抛弃了。如今 WebView 已经可以在需要的时候自动同步 cookie 了，所以不再需要创建 CookieSyncManager 类的对象来进行强制性的同步 cookie 了。现在只需要获得 CookieManager 的对象将 cookie 设置进去就可以了。

**前提：**

从服务器的返回头中取出 cookie 根据Http请求的客户端不同，获取 cookie 的方式也不同，请自行获取。

##### 1、客户端通过以下代码设置cookie，如果两次设置相同，会覆盖上一次的。

/\*\*

\* 将cookie设置到 WebView

\* @param url 要加载的 url

\* @param cookie 要同步的 cookie

\*/public static void syncCookie(String url,String cookie) {

if (Build.VERSION.SDK\_INT < Build.VERSION\_CODES.LOLLIPOP) {

CookieSyncManager.createInstance(context);

}

CookieManager cookieManager = CookieManager.getInstance();

cookieManager.setCookie(url, cookie);//如果没有特殊需求，这里只需要将session id以"key=value"形式作为cookie即可

}

**注意：**

1. 同步 cookie 要在 WebView 加载 url 之前，否则 WebView 无法获得相应的 cookie，也就无法通过验证。
2. cookie应该被及时更新，否则很可能导致WebView拿的是旧的session id和服务器进行通信。

##### 2、CookieManager会将这个Cookie存入该应用程序data/data/package\_name/app\_WebView/Cookies.db

##### 3、打开网页，WebView从数据库中读取该cookie值，放到http请求的头部，传递到服务器

/\*\*

\* 获取指定 url 的cookie

\*/public static String syncCookie(String url) {

CookieManager cookieManager = CookieManager.getInstance();

return cookieManager.getCookie(url);

}

##### 4、清除Cookie:

// 这个两个在 API level 21 被抛弃

CookieManager.getInstance().removeSessionCookie();

CookieManager.getInstance().removeAllCookie();

// 推荐使用这两个， level 21 新加的

CookieManager.getInstance().removeSessionCookies();// 移除所有过期 cookie

CookieManager.getInstance().removeAllCookies(); // 移除所有的 cookie

private void removeCookie(Context context) {

CookieManager.getInstance().removeAllCookies(new ValueCallback<Boolean>() {

@Override

public void onReceiveValue(Boolean value) {

// 清除结果

}

});

}

### WebView的方法

**前进、后退**

goBack()//后退

goForward()//前进

goBackOrForward(intsteps) //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

canGoForward()//是否可以前进

canGoBack() //是否可以后退

**清除缓存数据：**

clearCache(true);//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.

clearHistory()//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.

clearFormData()//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的状态：**

onResume() //激活WebView为活跃状态，能正常执行网页的响应

onPause()//当页面被失去焦点被切换到后台不可见状态，需要执行onPause动过， 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。

pauseTimers()//当应用程序被切换到后台我们使用了webview， 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。

resumeTimers()//恢复pauseTimers时的动作。

destroy()//销毁，关闭了Activity时，音乐或视频，还在播放。就必须销毁。

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();

**判断WebView是否已经滚动到页面底端 或者 顶端:**  
getScrollY() //方法返回的是当前可见区域的顶端距整个页面顶端的距离,也就是当前内容滚动的距离.  
getHeight()或者getBottom() //方法都返回当前WebView这个容器的高度  
getContentHeight()返回的是整个html的高度,但并不等同于当前整个页面的高度,因为WebView有缩放功能,所以当前整个页面的高度实际上应该是原始html的高度再乘上缩放比例.因此,更正后的结果,准确的判断方法应该是：

if (webView.getContentHeight() \* webView.getScale() == (webView.getHeight() + webView.getScrollY())) {

//已经处于底端

}

if(webView.getScrollY() == 0){

//处于顶端

}

### 避免WebView内存泄露的一些方式

1.可以将 Webview 的 Activity 新起一个进程，结束的时候直接System.exit(0);退出当前进程；  
启动新进程，主要代码： AndroidManifest.xml 配置文件代码如下

<activity

android:name=".ui.activity.Html5Activity"

android:process=":lyl.boon.process.web">

<intent-filter>

<action android:name="com.lyl.boon.ui.activity.htmlactivity"/>

<category android:name="android.intent.category.DEFAULT"/>

</intent-filter>

</activity>

在新进程中启动 Activity ，里面传了 一个 Url：

Intent intent = new Intent("com.lyl.boon.ui.activity.htmlactivity");

Bundle bundle = new Bundle();

bundle.putString("url", gankDataEntity.getUrl());

intent.putExtra("bundle",bundle);

startActivity(intent);

然后在 Html5Activity 的onDestory() 最后加上 System.exit(0); 杀死当前进程。

2.不能在xml中定义 Webview ，而是在需要的时候创建，并且Context使用 getApplicationgContext()，如下代码：

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.在 Activity 销毁的时候，可以先让 WebView 加载null内容，然后移除 WebView，再销毁 WebView，最后置空。  
代码如下：

@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();

}

### 返回键

返回上一次浏览的页面

public boolean onKeyDown(int keyCode, KeyEvent event) {

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

mWebView.goBack();

return true;

}

return super.onKeyDown(keyCode, event);

}

有一个非常不错的 Html5Activity 加载类帖出来：

package com.lyl.web;

import android.graphics.Bitmap;import android.os.Bundle;import android.os.Message;import android.support.v7.app.AppCompatActivity;import android.util.Log;import android.view.KeyEvent;import android.webkit.GeolocationPermissions;import android.webkit.WebChromeClient;import android.webkit.WebSettings;import android.webkit.WebView;import android.webkit.WebViewClient;

import com.lyl.test.R;

public class Html5Activity extends AppCompatActivity {

private String mUrl;

private LinearLayout mLayout;

private WebView mWebView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_web);

Bundle bundle = getIntent().getBundleExtra("bundle");

mUrl = bundle.getString("url");

Log.d("Url:", mUrl);

mLayout = (LinearLayout) findViewById(R.id.web\_layout);

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, ViewGroup.LayoutParams.MATCH\_PARENT);

mWebView = new WebView(getApplicationContext());

mWebView.setLayoutParams(params);

mLayout.addView(mWebView);

WebSettings mWebSettings = mWebView.getSettings();

mWebSettings.setSupportZoom(true);

mWebSettings.setLoadWithOverviewMode(true);

mWebSettings.setUseWideViewPort(true);

mWebSettings.setDefaultTextEncodingName("utf-8");

mWebSettings.setLoadsImagesAutomatically(true);

//调用JS方法.安卓版本大于17,加上注解 @JavascriptInterface

mWebSettings.setJavaScriptEnabled(true);

saveData(mWebSettings);

newWin(mWebSettings);

mWebView.setWebChromeClient(webChromeClient);

mWebView.setWebViewClient(webViewClient);

mWebView.loadUrl(mUrl);

}

@Override

public void onPause() {

super.onPause();

webView.onPause();

webView.pauseTimers(); //小心这个！！！暂停整个 WebView 所有布局、解析、JS。

}

@Override

public void onResume() {

super.onResume();

webView.onResume();

webView.resumeTimers();

}

/\*\*

\* 多窗口的问题

\*/

private void newWin(WebSettings mWebSettings) {

//html中的\_bank标签就是新建窗口打开，有时会打不开，需要加以下

//然后 复写 WebChromeClient的onCreateWindow方法

mWebSettings.setSupportMultipleWindows(false);

mWebSettings.setJavaScriptCanOpenWindowsAutomatically(true);

}

/\*\*

\* HTML5数据存储

\*/

private void saveData(WebSettings mWebSettings) {

//有时候网页需要自己保存一些关键数据,Android WebView 需要自己设置

mWebSettings.setDomStorageEnabled(true);

mWebSettings.setDatabaseEnabled(true);

mWebSettings.setAppCacheEnabled(true);

String appCachePath = getApplicationContext().getCacheDir().getAbsolutePath();

mWebSettings.setAppCachePath(appCachePath);

}

WebViewClient webViewClient = new WebViewClient(){

/\*\*

\* 多页面在同一个WebView中打开，就是不新建activity或者调用系统浏览器打开

\*/

@Override

public boolean shouldOverrideUrlLoading(WebView view, String url) {

view.loadUrl(url);

return true;

}

};

WebChromeClient webChromeClient = new WebChromeClient() {

//=========HTML5定位==========================================================

//需要先加入权限

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

//<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>

//<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>

@Override

public void onReceivedIcon(WebView view, Bitmap icon) {

super.onReceivedIcon(view, icon);

}

@Override

public void onGeolocationPermissionsHidePrompt() {

super.onGeolocationPermissionsHidePrompt();

}

@Override

public void onGeolocationPermissionsShowPrompt(final String origin, final GeolocationPermissions.Callback callback) {

callback.invoke(origin, true, false);//注意个函数，第二个参数就是是否同意定位权限，第三个是是否希望内核记住

super.onGeolocationPermissionsShowPrompt(origin, callback);

}

//=========HTML5定位==========================================================

//=========多窗口的问题==========================================================

@Override

public boolean onCreateWindow(WebView view, boolean isDialog, boolean isUserGesture, Message resultMsg) {

WebView.WebViewTransport transport = (WebView.WebViewTransport) resultMsg.obj;

transport.setWebView(view);

resultMsg.sendToTarget();

return true;

}

//=========多窗口的问题==========================================================

};

@Override

public boolean onKeyDown(int keyCode, KeyEvent event) {

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

mWebView.goBack();

return true;

}

return super.onKeyDown(keyCode, event);

}

@Override

protected void onDestroy() {

super.onDestroy();

if (mWebView != null) {

mWebView.clearHistory();

((ViewGroup) mWebView.getParent()).removeView(mWebView);

mWebView.loadUrl("about:blank");

mWebView.stopLoading();

mWebView.setWebChromeClient(null);

mWebView.setWebViewClient(null);

mWebView.destroy();

mWebView = null;

}

}

}