# （Android）WebView简要学习

## 前言

在Android中，如果需要用到网页的时候，那该如何使用呢。或者说原生应用在某些方面无法满足用户的要求，越来越多的混合开发产生了，使用android原生和h5页面整合也变成一种很不错的方式。

## 介绍

Android中有一种基于webkit引擎的高速浏览器，由它所封装的空间就是webview。

## 作用

1、显示web页面（浏览器不就是干这个的嘛）

2、使用html作为布局

3、js和android调用的中介

## 使用

WebView类

1. 加载网页几种方式

//方式1. 加载一个网页：  
webView.loadUrl("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");  
  
// 方式4： 加载 HTML 页面的一小段内容  
WebView.loadData(String data, String mimeType, String encoding)

1. 生命周期（状态）

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

1. 前进和后退

//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

1. 清理缓存

//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);  
  
//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()；  
  
//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()；

**WebSetting类**

主要是配置管理的功能。

常用配置方法：

//声明WebSettings子类  
 WebSettings webSettings = webView.getSettings();  
  
//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);  
// 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  
  
//支持插件  
webSettings.setPluginsEnabled(true);  
  
//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  
  
//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  
  
//其他细节操作  
webSettings.setCacheMode(WebSettings.*LOAD\_CACHE\_ELSE\_NETWORK*);

//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。

//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

**WebViewClient类**

主要是处理各种通知请求事件。

webview.setWebViewClient(new WebViewClient() {  
 @Override  
 public void onPageStarted(WebView view, String url, Bitmap favicon) {  
 super.onPageStarted(view, url, favicon);  
 //设定加载开始的操作  
 }  
  
 @Override  
 public void onPageFinished(WebView view, String url) {  
 super.onPageFinished(view, url);  
 //设定加载结束的操作  
 }  
  
 @Override  
 public void onLoadResource(WebView view, String url) {  
 super.onLoadResource(view, url);  
 //每个资源图片的加载都会调用  
 }  
  
 @Override  
 public void onReceivedError(WebView view, WebResourceRequest request, WebResourceError error) {  
 super.onReceivedError(view, request, error);  
 //加载服务异常的时候调用  
 }  
  
 @Override  
 public void onReceivedSslError(WebView view, SslErrorHandler handler, SslError error) {  
 super.onReceivedSslError(view, handler, error);  
 }  
  
});  
//5.1以上默认http和https不混合处理  
//开启  
if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*LOLLIPOP*) {  
 webview.getSettings().setMixedContentMode(WebSettings.*MIXED\_CONTENT\_ALWAYS\_ALLOW*);  
}

**WebChromeClient类**

主要是辅助webview处理js对话框等等。

主要用法如下;

webview.setWebChromeClient(new WebChromeClient(){  
 @Override  
 public void onProgressChanged(WebView view, int newProgress) {  
 super.onProgressChanged(view, newProgress);  
 //newProgress 当前的加载进度  
 }  
  
 @Override  
 public void onReceivedTitle(WebView view, String title) {  
 super.onReceivedTitle(view, title);  
 //获取网页中的标题  
 }  
  
 @Override  
 public boolean onJsAlert(WebView view, String url, String message, JsResult result) {  
 return super.onJsAlert(view, url, message, result);  
 //支持js中的警告框，自己定义弹出框 返回true  
 }  
  
 @Override  
 public boolean onJsConfirm(WebView view, String url, String message, JsResult result) {  
 return super.onJsConfirm(view, url, message, result);  
 //支持js中的确认框，自己定义弹出框 返回true  
 }  
  
 @Override  
 public boolean onJsPrompt(WebView view, String url, String message, String defaultValue, JsPromptResult result) {  
 return super.onJsPrompt(view, url, message, defaultValue, result);  
 //支持js中的输入框，自己定义弹出框 返回true  
 }  
});

注意注意：

1. 最好不在xml中定义webview，容易内存泄漏，最好在activity中创建使用applicationcontext作为上下文。
2. activity摧毁的时候，webview清空记录，然后摧毁，然后null。

## Android和js互相调用

中介就是webview。

1. Android调用js代码

**通过webview的loadUri（）方法**

webview.loadUrl("javascript:js代码”);

**简单方便，没有返回值。**

**通过WebView的evaluateJavascript（）**

webview.evaluateJavascript("javascript:js代码 ", new ValueCallback<String>() {  
 @Override  
 public void onReceiveValue(String s) {  
 //js返回结果  
 }  
});

**效率高，需要4.4以上版本。**

1. js调用Android代码

**（1）通过WebView的addJavascriptInterface（）进行对象映射**

class AndroidJs extends Object {  
 @JavascriptInterface//必须使用注解哦  
 public void print(String test) {  
 Toast.*makeText*(WebViewAcitivity.this, test, Toast.*LENGTH\_LONG*).show();  
 }  
}

在android中;

webview.addJavascriptInterface(new AndroidJs(), "androidjs");

androidjs为在js中调用的对象名称。

**（2）通过 WebViewClient 的shouldOverrideUrlLoading ()方法回调拦截 url**

Android通过 WebViewClient 的回调方法shouldOverrideUrlLoading ()拦截 url，解析该 url 的协议，如果检测到是预先约定好的协议，就调用Android相应的方法 。

js中代码：

document.location = "js://webview?你好 ";

Android中在webviewclient中重写shouldOverrideUrlLoading方法，其中url为返回的内容，解析一下即可回调Android方法。

**（3）通过 WebChromeClient 的onJsAlert()、onJsConfirm()、onJsPrompt()方法回调拦截JS对话框alert()、confirm()、prompt() 消息**

在webchromeclient类中重写拦截方法就可以。

对比：

|  |  |  |  |
| --- | --- | --- | --- |
| 方式 | 优点 | 缺点 | 场所 |
| 1 | 方便 | 4.2一下有漏洞 | 简单情况 |
| 2 | 不存在漏洞 | 复杂 | 不需要返回值 |
| 3 | 不存在漏洞 | 复杂 | 满足大部分情况 |

献上demo

https://github.com/xiaogoudandan/WilliamApp