转载请注明：http://blog.csdn.net/feather_wch/article/details/79364349

> 介绍Android中通过WebView与web页面交互的知识点。
> 分为三部分：
> 第一部分：Android中调用JS的代码
> 第二部分：JS调用Android中的代码

Android与JS交互

版本：2018/3/1-1

1-WebView的简单使用

打开网页

```
webview.setWebViewClient(new WebViewClient());
webview.setWebChromeClient(new WebChromeClient());
webview.loadUrl("https://www.baidu.com/");
webview.requestFocus();
```

需要网络权限AndroidManifest.xml:

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

2-Android中调用JS中的函数

1-直接调用html中js的方法

1、创建 `javascript.html` 文件，放置于 `src/main/assets` 目录下, JS代码如下:

```
// 文本名：javascript
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>Carson_Ho</title>

    // JS代码
    <script>
// Android需要调用的方法
   function callJS(){
      alert("Android调用了JS的callJS方法");
   }
</script>

</head>

</html>
```

2、Android中调用JS的函数

```
//1. 设置WebView
WebSettings settings = webview.getSettings();
settings.setJavaScriptEnabled(true);
settings.setJavaScriptCanOpenWindowsAutomatically(true);
settings.setSupportZoom(true);
//是否使用缓存
settings.setAppCacheEnabled(true);

/**=========================
 * 2. 需要支持js对话框
 *  webview只是载体，内容的渲染需要使用webviewChromClient类去实现
 *  通过设置WebChromeClient对象处理JavaScript的对话框
 *  设置响应js 的Alert()函数
 *=========================*/

webview.setWebChromeClient(new WebChromeClient() {
    //1. progress显示进度条
    @Override
    public void onProgressChanged(WebView view, int newProgress) {
        progressBar.setVisibility(View.VISIBLE);
        progressBar.setProgress(newProgress);
        if (newProgress == 100) {
            progressBar.setVisibility(View.GONE);
        }
        super.onProgressChanged(view, newProgress);
    }
    //2. 警告对话框
    @Override
    public boolean onJsAlert(WebView view, String url, String message, final JsResult result) {
        AlertDialog.Builder b = new AlertDialog.Builder(Main2Activity.this);
        b.setTitle("Alert");
        b.setMessage(message);
        b.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                result.confirm();
            }
        });
        b.setCancelable(false);
        b.create().show();
        return true;
    }

});
webview.setWebViewClient(new WebViewClient());
//3. 加载本地html
webview.loadUrl("file:///android_asset/javascript.html");
webview.post(new Runnable() {
    @Override
    public void run() {

        // 注意调用的JS方法名要对应上
        // 调用javascript的callJS()方法
        webview.loadUrl("javascript:callJS()");
    }
});
```

- 特别注意：JS代码调用一定要在 onPageFinished（） 回调之后才能调用，否则不会调用。

> onPageFinished()属于WebViewClient类的方法，主要在页面加载结束时调用

调用本地html文件(该html文件调用了两个js文件中的方法)

- 注意html和js文件的目录关系，需要根据html中的引入路径

本例中有三个文件：servicequery.html、contentloader.js、md5.js
html文件放置于 `src/main/assets` 目录下，且两个js文件根据引入路径放置于 `src/main/assets/js` 目录下

2-通过evaluateJavascript与JS进行交互

```
/**
 * evaluateJavascript()不会刷新页面，loadUrl会刷新页面
 */
webview.post(new Runnable() {
    @Override
    public void run() {
        //javascript与html文件对应
        webview.evaluateJavascript("javascript:callJS()", new ValueCallback<String>() {
            @Override
            public void onReceiveValue(String value) {
                //此处为JS返回的结果
            }
        });
    }
});
```

- 建议进行混合使用

```
final int version = Build.VERSION.SDK_INT;
// Android 4.4 版本以及以上版本时，使用evaluateJavascript(),否则使用loadUrl()
if (version < 18) {
    webview.loadUrl("javascript:callJS()");
} else {
    webview.evaluateJavascript("javascript:callJS()", new ValueCallback<String>() {
        @Override
        public void onReceiveValue(String value) {
            //此处为 js 返回的结果
        }
    });
}
```

3-JS通过WebView调用Android代码

JS调用Android代码有三种方法

1. 通过 WebView 的 addJavascriptInterface() 进行对象映射
2. 通过 WebViewClient 的 shouldOverrudeUrlLoading() 方法回调拦截 url
3. 通过 WebChromeClient 的 onJsAlert()、onJsConfirm()、onJsPrompt() 方法回调拦截JS对话框 alert()、confirm()、prompt() 消息

1- addJavascriptInterface() 进行对象映射

1-js代码

//jsCallAndroid.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Carson</title>
    <script>
         function callAndroid(){
        // 由于对象映射，所以调用test对象等于调用Android映射的对象
            jsObject.hello("js调用了android中的hello方法");
         }
      </script>
</head>
<body>
//点击按钮则调用callAndroid函数
<button type="button" id="button1" onclick="callAndroid()"></button>
</body>
</html>
```

2-Android对象：用于和js对象映射

```
public class AndroidtoJs extends Object{
    // 定义JS需要调用的方法，必须使用注解@JavascriptInterface
    @JavascriptInterface
    public void hello(String msg){
        System.out.println(msg);
    }
}
```

3-通过WebView设置Android类和JS对象映射关系

```
WebSettings settings = webview.getSettings();
//1. 设置于JS交互的权限
settings.setJavaScriptEnabled(true);
//2. 将Java对象映射到JS对象
webview.addJavascriptInterface(new AndroidtoJs(), "jsObject");
//3. 加载JS代码
webview.loadUrl("file:///android_asset/jsCallAndroid.html");
```

- 使用简单方便
- 存在严重的漏洞：https://www.jianshu.com/p/3a345d27cd42 -Android 4.2版本开始通过 `@JavascriptInterface` 规避该漏洞

实例二: 在JS中调用Android中的对话框

```
public class WebViewJSActivity extends Activity {

    @Bind(R.id.progress_bar)
    ProgressBar progressBar;
    @Bind(R.id.webview)
    WebView webview;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_web_view_js);
        ButterKnife.bind(this);

        WebSettings settings = webview.getSettings();
        //1. 设置于JS交互的权限
        settings.setJavaScriptEnabled(true);
        //2. 将Java对象映射到JS对象
        webview.addJavascriptInterface(WebViewJSActivity.this, "jsObject");
        //3. 加载JS代码
        webview.loadUrl("file:///android_asset/jsCallAndroid.html");
    }

    //用于在Web的JS中创建Android的dialog
    @JavascriptInterface
    public void displayDialog(String msg){
        AlertDialog.Builder builder = new AlertDialog.Builder(WebViewJSActivity.this);
        builder.setTitle("Alert").setMessage(msg).setCancelable(false).setPositiveButton(android.R.string.ok, new DialogInterface.OnClic
            @Override
            public void onClick(DialogInterface dialog, int which) {
            }
        }).create().show();
    }
}
```

jsCallAndroid.xml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Carson</title>
    <script>
         function callAndroid(){
        // 由于对象映射，所以调用test对象等于调用Android映射的对象
            jsObject.displayDialog("我是JS代码中的信息");
         }
      </script>
</head>
<body>
<input type="button"
        id="button1"
        onclick="callAndroid()"
        style="width:250px;height:50px"
        value="点击按钮则调用Android中的警告框"></input>
</body>
</html>
```

2-通过WebViewClient的 `shouldOverrideUrlLoading` 进行url拦截

1-js中约定url协议

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Carson</title>
    <script>
         function callByShouldOverrideUrlLoading(){
            /*约定的url协议为：js://webview?arg1=111&arg2=222*/
            document.location = "js://webview?arg1=111&arg2=222";
         }

      </script>
</head>
<body>
<input type="button"
        id="button1"
        onclick="callByShouldOverrideUrlLoading()"
        style="width:250px;height:50px"
        value="点击按钮则调用Android中的警告框"></input>
</body>
</html>
```

2-Android中在 `shouldOverrideUrlLoading` 进行解析拦截

`shouldOverrideUrlLoading` 会被回调

```
webview.setWebViewClient(new WebViewClient(){
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
       /*==============================================*
        * 1. 根据协议的参数，判断是否是所需要的url
        * 一般根据scheme（协议格式） & authority（协议名）判断（前两个参数）
        * 假定传入进来的 url = "js://webview?arg1=111&arg2=222"（同时也是约定好的需要拦截的）
        *==============================================*/

        Uri uri = Uri.parse(url);
        // 2. 如果url的协议 = 预先约定的 js 协议，就解析往下解析参数
        if ( uri.getScheme().equals("js")) {

            // 3. 如果 authority  = 预先约定协议里的 webview，即代表都符合约定的协议
            if (uri.getAuthority().equals("webview")) {

                // 4. 执行JS所需要调用的逻辑
                displayDialog("js调用了Android的方法: 通过shouldOverrideUrlLoading");

                // 5. 可以在协议上带有参数并传递到Android上
                HashMap<String, String> params = new HashMap<>();
                Set<String> collection = uri.getQueryParameterNames();
            }

            return true;
        }
        return super.shouldOverrideUrlLoading(view, url);
    }
});


//用于在Web的JS中创建Android的dialog
@JavascriptInterface
public void displayDialog(String msg){
    AlertDialog.Builder builder = new AlertDialog.Builder(WebViewJSActivity.this);
    builder.setTitle("Alert").setMessage(msg).setCancelable(false).setPositiveButton(android.R.string.ok, new DialogInterface.OnClickLis
        @Override
        public void onClick(DialogInterface dialog, int which) {
        }
    }).create().show();
}
```

JS中需要获得Android方法执行的返回值?

通过loadUrl去执行JS中的方法将返回值作为参数传递进去-比较繁琐复杂

```
// Android: MainActivity.java
mWebView.loadUrl("javascript:returnResult(" + result + ")");

// JS: javascript.html
function returnResult(result){
    alert("result is" + result);
}
```

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

JS中三种常用对话框

方法	作用	返回值	备注
alert()	弹出警告框	没有	在文本加入\n可以换行
confirm()	弹出确认框	两个返回值	1. 返回布尔值 2. 通过该值可判断点击是确认还是取消：true-确认，false-取消
prompt()	弹出输入框	任意设置返回值	1. 点击“确认”:返回输入框中的值 2. 点击“取消”:返回null

原理

Android中通过 `WebChromeClient` 的 `onJsAlert()\onJsConfirm()\onJsPrompt()` 方法回调分别拦截JS对话框。

实例: 在JS的“输入框”中输入数据，并将输入的数据在Android中通过警告框显示出来

JS代码:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Carson</title>
    <script>
         function clickPrompt(){
            //1. prompt输入框获取输入信息
            var result=prompt("js://demo?arg1=111&arg2=222");
            //2. 将“输入数据”通过Alert发送给Android
            alert(result);
         }
      </script>
</head>
<body>
<input type="button"
        id="button1"
        onclick="clickPrompt()"
        style="width:250px;height:50px"
        value="点击按钮则调用Android中的警告框"></input>
</body>
</html>
```

Android中通过WebChromeClient进行拦截:

```
webview.setWebChromeClient(new WebChromeClient(){
            //可以在Android中拦截输入框，这里不进行拦截
            @Override
            public boolean onJsPrompt(WebView view, String url, String message, String defaultValue, JsPromptResult result) {
                //1. 获取prompt中定义好的协议，这里是“js://demo?arg1=111&arg2=222”
                Uri uri = Uri.parse(message);
                if(uri.getScheme().equals("js")){
                    if(uri.getAuthority().equals("demo")){
                      //2. 可以进行一些处理操作
                      //System.out.println("js调用了Android的方法");

                      /**
                       * 协议中传递参数：这里我没有进行处理
                       */
                      HashMap<String, String> params = new HashMap<>();
                      Set<String> collection = uri.getQueryParameterNames();

                      //result.confirm("该String代表消息框的返回值(输入值)");
                      //return true;
                    }
                }
                //3. 这里直接弹出输入框，让用户输入数据
                return super.onJsPrompt(view, url, message, defaultValue, result);
            }

            //4. 拦截到用户输入的数据并且进行显示（本质是拦截Alert）
            @Override
            public boolean onJsAlert(WebView view, String url, String message, JsResult result) {
                displayDialog("message=" + message);
                result.confirm();
                return true;
            }
        });
```

对话框的代码:

```
//用于在Web的JS中创建Android的dialog
@JavascriptInterface
public void displayDialog(String msg){
    AlertDialog.Builder builder = new AlertDialog.Builder(WebViewJSActivity.this);
    builder.setTitle("Alert").setMessage(msg).setCancelable(false).setPositiveButton(android.R.string.ok, new DialogInterface.OnClickLis
        @Override
        public void onClick(DialogInterface dialog, int which) {
        }
    }).create().show();
}
```

4-三种JS调用Android代码的总结

调用方法	优点	缺点	使用场景
addJavascriptInterface():对象映射	方便简洁	Android4.2以下存在漏洞	Android4.2以上
WebViewClient的shouldOverrideUrlLoading:拦截Url	不存在漏洞	使用复杂； 需要进行协议约束；	不需要返回值情况下的互动场景 （IOS中主要使用该方法）
WebChromeClient的onJsAlert等方法回调拦截Js的对话框消息	不存在漏洞	使用复杂； 需要进行协议约束；	能满足大多数情况下的互调场景