本文主要对WebView进行介绍，包括webView 4个可以定制的点、设置WebView back键响应、控制网页的链接仍在webView中跳转、显示页面加载进度、处理https请求、利用addJavascriptInterface实现[android](http://bbs.189works.com/)程序和[java](http://www.189works.com/tech/devlodoc/java/)script交互等等  
  
WebView基于webkit引擎展现web页面的控件，使用前需要在Android Manifest file中配置internet访问权限，否则提示页面无法访问  
Xml代码  [url=][/url]

1. <USES-PERMISSION  
   [android](http://www.189works.com/tech/devloplat/Android/):name="android.permission.INTERNET"  
   />

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。  
WebView有几个可以定制的点：  
（1）设置[WebChromeClient](http://developers.androidcn.com/reference/Android/webkit/WebChromeClient.html)子类，**WebChromeClient会在一些影响浏览器ui交互动作发生时被调用**，比如WebView关闭和隐藏、页面加载进展、js确认框和警告框、js加载前、js操作超时、webView获得焦点等等，详见[WebChromeClient](http://developers.androidcn.com/reference/Android/webkit/WebChromeClient.html)  
  
（2）设置[WebViewClient](http://developers.androidcn.com/reference/Android/webkit/WebViewClient.html)子类，**WebViewClient会在一些影响内容喧嚷的动作发生时被调用**，比如表单的错误提交需要重新提交、页面开始加载及加载完成、资源加载中、接收到http认证需要处理、页面键盘响应、页面中的url打开处理等等，详见[WebViewClient](http://developers.androidcn.com/reference/Android/webkit/WebViewClient.html)  
  
（3）设置[WebSettings](http://developers.androidcn.com/reference/Android/webkit/WebSettings.html)类，**其中包含多项配置**。WebSettings用来对WebView的配置进行配置和管理，比如是否可以进行文件操作、缓存的设置、页面是否支持放大和缩小、是否允许使用数据库api、字体及文字编码设置、是否允许js脚本运行、是否允许图片自动加载、是否允许数据及密码保存等等，详见[WebSettings](http://developers.androidcn.com/reference/Android/webkit/WebSettings.html)  
  
（4）设置[addJavascriptInterface](http://developers.androidcn.com/reference/Android/webkit/WebView.html#addJavascriptInterface%28java.lang.Object,%20java.lang.String%29)方法，**将java对象绑定到webView中，以方便从页面js中控制java对象**， 实现用本地java代码和html页面进行交互，甚至可以进行页面自动化。但如此做存在安全隐患，所以若设置了此方法，请确保webView的代码都是自 己完成，详细使用addJavascriptInterface进行自动化见本文5使用addJavascriptInterface完成和js交互  
  
**1、back键控制网页后退**  
Activity默认的back键处理为结束当前Activity，webView查看了很多网页后，希望按back键返回上一次浏览的页面，这个时候我们就需要覆盖onKeyDown函数，告诉他如何处理，如下：  
Java代码  [url=][/url]

1. public  
   boolean onKeyDown(int keyCode, KeyEvent event) {
2. if (webView.canGoBack() && event.getKeyCode() == KeyEvent.KEYCODE\_BACK && event.getRepeatCount() == 0) {
3. webView.goBack();
4. return  
   true;
5. }
6. return  
   super.onKeyDown(keyCode, event);
7. }

其中webView.canGoBack()在webView含有一个可后退的浏览记录时返回true  
webView.goBack();表示返回webView的上次访问页面  
  
**2、当前网页的链接仍在webView中跳转**  
Java代码  [url=][/url]

1. webView.setWebViewClient(new WebViewClient() {
2. @Override
3. public  
   boolean shouldOverrideUrlLoading(WebView view, String url) {
4. view.loadUrl(url);
5. return  
   true;
6. }
7. });

shouldOverrideUrlLoading表示当前webView中的一个新url需要加载时，给当前应用程序一个处理机会，如果没有重写此函 数，webView请求ActivityManage选择合适的方式处理请求，就像弹出uc和互联网让用户选择浏览器一样。重写后return true表示让当前程序处理，return false表示让当前webView处理  
Xml代码  [url=][/url]

1. Give the host application a chance to take over the control when a new url is about to be loaded in the current WebView. If WebViewClient is not provided, by default WebView will ask Activity Manager to choose the proper handler for the url. If WebViewClient is provided, return true means the host application handles the url, while return false means the current WebView handles the url.

参考：  
[http://developer.android.com/reference/android/webkit/WebViewClient.html#shouldOverrideUrlLoading(Android.webkit.WebView, java.lang.String)](http://developer.android.com/reference/android/webkit/WebViewClient.html#shouldOverrideUrlLoading%28Android.webkit.WebView,%20java.lang.String%29)  
  
问题分析：  
今天在用webView load腾讯的OAuth页面<http://open.t.qq.com/cgi-bin/authorize>  
时，突然弹出选择uc还是互联网浏览器的确认框，很奇怪，其他页面都能正常load，而这个却无法在webView中load，后来用火狐看了下是因为load这个页面的时候会重定向到另外一个页面<http://open.t.qq.com/oauth_html/login.php?oauth_token=xxx&type=0>，而webView没有设置shouldOverrideUrlLoading，导致url交给了ActivityManage去处理，解释如上，后添加后可正常load页面  
  
**3、页面加载进度**  
代码如下  
Java代码  [url=][/url]

1. webView.setWebChromeClient(new WebChromeClient() {
2. public  
   void onProgressChanged(WebView view, int progress) {
3. setTitle("页面加载中，请稍候..." + progress + "%");
4. setProgress(progress \* 100);
5. if (progress == 100) {
6. setTitle(R.string.app\_name);
7. }
8. }
9. });

onProgressChanged通知应用程序当前页面加载的进度  
progress  
表示当前页面加载的进度，为1至100的整数  
参考：  
[http://developer.android.com/reference/android/webkit/WebChromeClient.html#onProgressChanged(android.webkit.WebView, int)](http://developer.android.com/reference/android/webkit/WebChromeClient.html#onProgressChanged%28Android.webkit.WebView,%20int%29)  
  
**4、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：  
Java代码  [url=][/url]

1. webView.setWebViewClient(new WebViewClient() {
2. @Override
3. public  
   void onReceivedSslError(WebView view, SslErrorHandler handler, SslError error) {
4. handler.proceed();
5. //            handler.cancel();
6. //            handler.handleMessage(null);
7. }
8. });

onReceivedSslError为webView处理ssl证书设置  
其中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  
参考：  
[http://developer.android.com/reference/android/webkit/WebViewClient.html#onReceivedSslError(android.webkit.WebView, android.webkit.SslErrorHandler, android.net.http.SslError)](http://developer.android.com/reference/android/webkit/WebViewClient.html#onReceivedSslError%28android.webkit.WebView,%20android.webkit.SslErrorHandler,%20Android.net.http.SslError%29)  
  
**5、使用addJavascriptInterface完成和js交互**  
**5.1 javascript中调用java对象及方法**  
设置webView的addJavascriptInterface方法，该方法有两个参数，第一个参数为被绑定到js中的类实例，第二个参数为在js中暴露的类别名，在js中引用java对象就是用这个名字  
Java代码  [url=][/url]

1. ClassBeBindedToJS classBeBindedToJS = new ClassBeBindedToJS();
2. webView.addJavascriptInterface(classBeBindedToJS, "classNameBeExposedInJs");

实现绑定到js的类，其中的javaMethod方法我们将在页面前端js中调用，用于返回一段内容。showHtml用来使用AlertDialog显示一段html代码，稍后将使用到  
Java代码  [url=][/url]

1. private  
   class ClassBeBindedToJS{
2. public  
   void showHtml(String html){
3. new AlertDialog.Builder(UpdateStatusActivity.this)
4. .setTitle("HTML").setMessage(html)
5. .setPositiveButton(android.R.string.ok, null)
6. .setCancelable(false).create().show();
7. }
8. public String javaMethod() {
9. return  
   "use java method";
10. }
11. };

如此我们就可以在前端调用java对象，如下：  
Html代码  [url=][/url]

2. <DIV  
   id="displayDiv">Test page.

         <INPUT  
type="button"  
value="http://www.cnblogs.com/Developers/archive/2012/03/22/use java object"  
onclick="document.getElementById('displayDiv').innerHTML=classNameBeExposedInJs.javaMethod()"  
/>





这段html作用为点击button按钮，改变div内容为java对象方法中的内容，其中classNameBeExposedInJs为java对象在js中的别名，javaMethod为java对象的方法  
  
**5.2 java中调用js方法**  
用webView的loadUrl实现，比如现在我们想在页面加载完成后调用js中的hello函数，实现如下：  
Java代码  [url=][/url]

1. webView.setWebViewClient(new WebViewClient() {
2. @Override
3. public  
   void onPageFinished(WebView webView, String url){
4. webView.loadUrl("javascript:hello()");
5. }
6. });

怎么样，很简单吧^\_^,直接loadUrl("javascript:hello()")就可以了   
利用这一点我们可以很方便的实现WebView的自动化，如网页自动化登录。举个栗子，实现<http://m.renren.com/> 的自动登录啊，代码如下  
Java代码  [url=][/url]

1. webView.getSettings().setJavaScriptEnabled(true);
2. webView.loadUrl("http://m.renren.com");
3. webView.requestFocus();
4. ClassBeBindedToJS classBeBindedToJS = new ClassBeBindedToJS();
5. webView.addJavascriptInterface(classBeBindedToJS, "classNameBeExposedInJs");
6. webView.setWebViewClient(new WebViewClient() {
7. @Override
8. public  
   void onPageFinished(WebView webView, String url){
9. webView.loadUrl("javascript:document.getElementsByName('email')[0].value='http://www.cnblogs.com/Developers/archive/2012/03/22/userName'");
10. webView.loadUrl("javascript:document.getElementsByName('password')[0].value='http://www.cnblogs.com/Developers/archive/2012/03/22/userPassword'");
11. webView.loadUrl("javascript:document.getElementsByName('login')[0].click()");
12. }
13. });

从中可以看出先加载http://m.renren.com页面，在页面加载完成onPageFinished后，调用js实现自动登录  
其中三句js分别表示设置用户名文本框内容、设置用户密码文本框内容、点击登录按钮，如此登录成功喽  
  
**5.3 java和js混用实现一些效果**  
**5.3.1 WebView用AlertDialog显示页面的html代码**  
之前5.1中的showHtml方法现在派上用场啦，在页面加载完成后显示页面html内容，如下：  
Java代码  [url=][/url]

1. webView.setWebViewClient(new WebViewClient() {
2. @Override
3. public  
   void onPageFinished(WebView webView, String url){
4. webView.loadUrl("javascript:window.classNameBeExposedInJs.showHtml(document.getElementsByTagName('html')[0].innerHTML);");
5. }
6. });

其中webView.loadUrl表示调用页面中的js，而页面中的js window.classNameBeExposedInJs.showHtml调用了程序中的java对象的方法，方法参数为一段js。  
首先webView会执行这一段js document.getElementsByTagName('html')[0].innerHTML。含义即为取得页面中html标记的innerHTML，及网页主要内容；  
然后将得到的网页内容当作字符串参数传入showHtml方法  
最后调用showHtml函数实现，即用AlertDialog显示字符串，如此得到了整个网页的大部分html代码  
  
**5.3.2 WebView显示当前页面的url地址**  
获得WebView当前页面url使用webView.getUrl()即可，如下：  
Java代码  [url=][/url]

1. webView.loadUrl("javascript:window.classNameBeExposedInJs.showHtml('" + webView.getUrl() + "');");