# Android Html5 开发

**Evan | 汤飞**

2013.04.17

www.eoe.cn

# Android应用开发新路线

- HTML5未来发展趋势
- 利用HTML5开发Android应用程序
- HTML5开发的特点

# HTML5未来发展趋势

- 移动优先
- 响应式设计&自动变化的屏幕尺寸
- 设备访问
- 离线缓存
- 开发工具的成熟

# 移动优先

- 从如今层出不穷的移动应用就知道，在这个智能手机和将平板电脑大爆炸的时代，移动优先已成趋势，不管是开发什么，都以**移动**为主。
- 此前一直困扰移动领域的问题就是开发Web应用还是原生应用。而如今，我们看见一些大型企业如《金融时报》在冲着HTM L5进军移动市场过程中，从App Store撤掉iPad原生应用而开发Web应用，同样表现出色。
- 许多游戏开发商也将在移动Web应用中扮演中重要角色，移动Web应用优先的趋势将会持续到移动设备统治信息处理领域。其实用户根本不在乎你用什么工具开发了什么应用，不管是Web应用还是原生应用，只要好用就可以了。

# 响应式设计&自动变化的屏幕尺寸

- 在HTML 5真的改变移动开发平台之前，必须要迈出重要一步，那就是“响应式设计”，也就是屏幕可以根据内容而自动调整大小。
- 响应式设计最好的一个例子BostonGlobe.com（观看视频），其屏幕能够根据任何内容而调整尺寸大小，在访问过其开发商Filament Group后才了解到，响应式设计也并非易事，一些基本概念设计必须从头开始，比如处理媒体库的RespondJS，而且处理来自第三方的图片和广告也是恼人的问题。
- 要想做好响应式设计，就必须洞悉内容与屏幕之间的反馈关系，一家来自硅谷的响应式设计公司ZURB称，其实在过去的16年中，开发商就意识到响应式设计就要完全离开“流”，转而注重内容是如何在网页和移动设备中被处理的，这一过程还在继续，HTML 5会让它最终成为可能。

# 设备访问

- 消除Web应用与原生应用界限的最大障碍就是浏览器访问移动设备基本特性的能力，比如照相机，通讯录，日历，加速器等，利用HTML5实现此能力方面，appMobi算是行业翘楚，现在也开源了所有API。Mozilla也一直在努力通过移动浏览器Fennec来将强设备访问能力。
- 对许多**移动开发**商来说，提高设备访问能力是HTML5最令人激动的革新，这意味着Web应用能够登陆移动设备而无需做任何PhoneGap式打包，游戏开发商当然最开心，因为某些特性对他们来说是封锁的，比如能整合到游戏中的加速器。
- 这就开启了另一个可能的世界，比如能与云更好地整合（这有利于应用内购买，消息推送等）并提高游戏可玩性，有了HTML5这个平台，开发商可以不再依赖于Java语言，CSS3，HTML及其它程序语言。

# 离线缓存

- 这个概念相当新潮，离线情况下，app也能照常运作，算是HTML 5充满魔力的一面，最好的离线缓存例子就是亚马逊Kindle的云阅读器，可以通过Firefox6以上版本，Chrome11以上版本，Safari5以上版本及iOS4以上版本浏览器将内容同步到所有Kindle系列设备，并能记忆用户在kindle图书馆的一切。
- 亚马逊就这么实现了离线使用Web应用，许多专家人声称原生应用的末日即将到来，因为Web应用的使用变得简单，无摩擦，适用于任何一个平台或者无需平台。

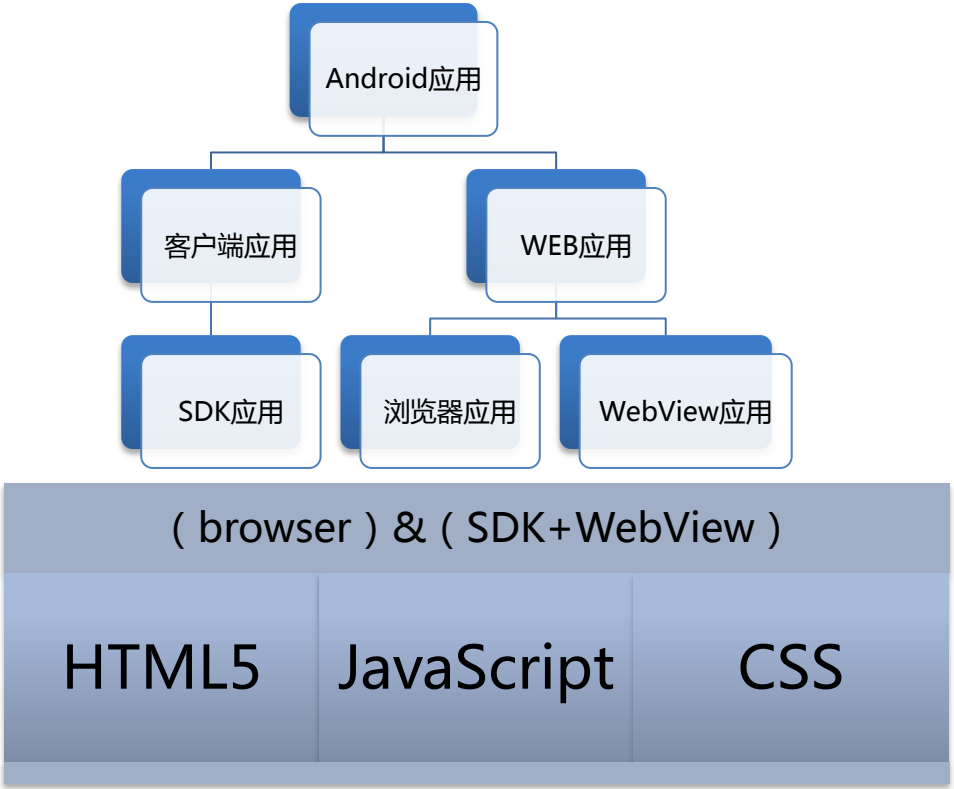
# 开发工具的成熟

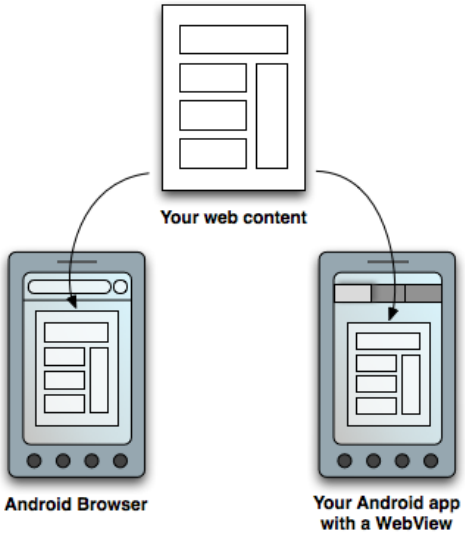
- 在工具方面，除了appMobi提供的工具包以外，还有Sencha及Appcelerator提供的框架及IDE供应用开发商们使用，虽然这些工具现在算不上成熟，也不如Android和iOS上的开发商框架及工具那般简单强大，但至少它们在演进，将会变得越来越好用。

# HTML5与Android融合

- Android的HTML5应用程序概述
- 如何适配多分辨率的Android设备？
- 如何在Android中构建HTML5应用程序？
- 如何结合PhoneGap进行开发？

# Android的HTML5应用程序概述



# 适配多分辨率的Android设备

Android设备的多分辨率？

- 物理分辨率
- 视窗大小与WEB页面比例
- 屏幕密度

Android浏览器加载WEB页面时，如果用户没有禁止启用”预览模式“，那么将默认为“预览模式“，通常会缩小WEB页面。而当页面在WebView中显示时，会采用”完全载入“的方式，即保证WEB页面的原始大小。

怎么办？

- viewport属性
- 用CSS控制设备密度
- 用JavaScript控制设备密度

设备屏幕的密度是基于屏幕的分辨率（由每英寸所包含的点数（dpi））定义的。Android支持三种类别的屏幕密度：低（ldpi），中（mdpi）和高（hdpi）。与中等密度屏幕相比，低密度屏幕每英寸像素较少，高密度屏幕每英寸像素较多。默认情况下，Android浏览器和 WebView是针对中等密度的屏幕。Android浏览器和WebView 在高密度屏幕上将Web页面缩放约1.5倍（因为中等密度屏幕像素较小），而在低密度屏幕上将Web页面缩放约0.75倍（因为中等密度屏幕像素大）。

# viewport属性的应用

- viewport需要放置在HTML的<meta>标签中，在<meta>标签的 content属性中，就可以定义多个视窗特性。包括视窗的宽度、高度、缩放比例，目标设备密度等，但是，需要注意每个视窗属性必须有逗号隔开。

```
<head>
    <title>Exmaple</title>
    <meta name="viewport" content="width=device-width,user-scalable=no"/>
</head>
```

```
<meta name="viewport"
      content="
            height = [pixel_value | device-height] ,
            width = [pixel_value | device-width ] ,
            initial-scale = float_value ,
            minimum-scale = float_value ,
            maximum-scale = float_value ,
            user-scalable = [yes | no] ,
            target-densitydpi = [dpi_value | device-dpi |
                                 high-dpi | medium-dpi | low-dpi]
      " />
```

# viewport属性的应用

**width**
控制 viewport 的大小，可以指定的一个值或者特殊的值，如 device-width 为设备的宽度（单位为缩放为 100% 时的 CSS 的像素）。

**height**
和 width 相对应，指定高度。

**target-densitydpi**
一个屏幕像素密度是由屏幕分辨率决定的，通常定义为每英寸点的数量（dpi）。Android支持三种屏幕像素密度：低像素密度，中像素密度，高像素密度。一个低像素密度的屏幕每英寸上的像素点更少，而一个高像素密度的屏幕每英寸上的像素点更多。Android Browser和WebView默认屏幕为中像素密度。

**initial-scale**
初始缩放。即页面初始缩放程度。这是一个浮点值，是页面大小的一个乘数。例如，如果你设置初始缩放为“1.0”，那么，web页面在展现的时候就会以target density分辨率的1:1来展现。如果你设置为“2.0”，那么这个页面就会放大为2倍。

**maximum-scale**
最大缩放。即允许的最大缩放程度。这也是一个浮点值，用以指出页面大小与屏幕大小相比的最大乘数。例如，如果你将这个值设置为”2.0”，那么这个页面与target size相比，最多能放大2倍。

**user-scalable**
用户调整缩放。即用户是否能改变页面缩放程度。如果设置为yes则是允许用户对其进行改变，反之为no。默认值是yes。如果你将其设置为no，那么minimum-scale 和 maximum-scale都将被忽略，因为根本不可能缩放。

所有的缩放值都必须在0.01–10的范围之内。

(设置屏幕宽度为设备宽度，禁止用户手动调整缩放)

```
 <meta name="viewport" content="width=device-width,user-scalable=no" />
```

(设置屏幕密度为高屏，中屏，低屏自动缩放，禁止用户手动调整缩放)

```
<meta name="viewport" content="width=device-width,target-densitydpi=high-dpi,initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no"/>
```

# CSS控制设备密度

- Android浏览和WebView支持CSS媒体性能（webkit-device-pixel-ratio），允许指定屏幕密度创建一些样式CSS媒体性能。该值应该是“0.75”，“1”或“1.5”，它们分别表示对于低、中、高密度屏幕的设备。

下面为每种密度创建独立的样式：

```
<link rel="stylesheet" media="screen and (-webkit-device-pixel-ratio: 1.5)" href="hdpi.css" />
<link rel="stylesheet" media="screen and (-webkit-device-pixel-ratio: 1.0)" href="mdpi.css" />
<link rel="stylesheet" media="screen and (-webkit-device-pixel-ratio: 0.75)" href="ldpi.css" />
```

在一个样式表中，指定不同样式：

```
#header {background:url(medium-density-image.png);}
@media screen and (-webkit-device-pixel-ratio: 1.5) {
    // CSS for high-density screens
    #header {
           background:url(high-density-image.png);
    }
}
@media screen and (-webkit-device-pixel-ratio: 0.75) {
    // CSS for low-density screens
    #header {background:url(low-density-image.png);}
}
```

# JavaScript控制设备密度

- Android浏览器和WebView支持查询当前设备密度的DOM特性（window.devicePixelRatio），该值指定用于当前设备按比例缩放的系数。例如，值为“1.0”，则说明设备是中等密度，并且默认页面不进行缩放；如果该值是“1.5”，那么，设备是高密度设备，并且默认页面调整1.5x（倍）；如果该值是“0.75”，那么，设备是低密度设备，并且默认页面调整0.75x（倍）。

如何使用JavaScript查询设备密度：

```
if (window.devicePixelRatio == 1.5) {
    alert("This is a high-density screen");
} else if (window.devicePixelRation == 0.75) {
    alert("This is a low-density screen");
}
```

# 在Android中构建HTML5应用程序

- 使用WebView在Android中构建Web应用
- 处理页面导航
- 浏览网页历史记录
- Android与JavaScript交互

# Android WebView应用

WebView 类是Android View类的扩展，它允许Web页面作为Activity布局的一部分显示。它不包括完整Web浏览器的任何功能，如导航控制或地址栏。默认情况下WebView 所能做的就是显示一个网页。

添加WebView到应用程序中:

```
<?xml version="1.0" encoding="utf-8"?>
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+id/webview  “
            …
/>
```

要在WebView中加载Web页面，使用loadUrl()

```
WebView  mWebView = (WebView) findViewById(R.id.webview);
mWebView.loadUrl("http://www.example.com");
```

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

# 处理页面导航

在WebView中，当用户从Web页面里点击一个链接，在Android中，默认行为是启动一个应用程序来处理URL。 通常，默认Web浏览器打开并加载这个目的URL。 但是，您可以为您的WebView忽略此默认行为 ，由 WebView 打开所有链接。 然后，通过WebView，您可以运行用户向前、向后浏览他们的Web页面的历史 。

```
private class MyWebViewClient extends WebViewClient {
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        if (Uri.parse(url).getHost().equals("www.example.com")) {
            return false;
        }
        return true;
    }
}
```

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.setWebViewClient(new MyWebViewClient());
```

shouldOverrideUrlLoading
shouldOverrideKeyEvent
onReceivedHttpAuthRequest
onReceiveError
onReceivedSslError
onLoadResource
onPageStarted
onPageFinished

# Android与JavaScript交互

- 如果您计划使用JavaScript将Web页面加载到WebView 中，您就必须为您的 WebView 启用 JavaScript。一旦启用JavaScript，您就可以在您的应用程序与您的JavaScript代码之间建立接口。
- 默认情况下，在WebView中，JavaScript是禁用的。您可以通过将WebSettings附加到您的WebView中来启用JavaScript。你可以用getSettings()检索WebSettings，然后用setJavaScriptEnabled() 启动JavaScript。

```
WebView myWebView = (WebView) findViewById(R.id.webview);
WebSettings webSettings = myWebView.getSettings();
webSettings.setJavaScriptEnabled(true);
```

Android与JS通信实例演示:

- 在Android中处理JS的警告、对话框等；
- 在JS中调用Android接口；
- 在Android调用JS函数。

# 在Android中处理JS的警告、对话框

- 在Android中处理JS的警告，对话框等需要对WebView设置WebChromeClient对象

```
webView.setWebChromeClient(new WebChromeClient(){
        public boolean onJsAlert(WebView view, String url, String message, final
                                        JsResult result) {
                //do something...
        };
        public boolean onJsConfirm(WebView view, String url, String message,
                                                final  JsResult result) {
                //do something...
        }
        public void onProgressChanged(WebView view, int newProgress) {
                //do something...
        }
        public void onReceivedTitle(WebView view, String title) {
                //do something...
        }
 });
```

# 在JS中调用Android接口

- 首先 需要在Android程序中建立接口

```
final class InJavaScript {
        public void runOnAndroidJavaScript(final String str) {
                //do something...
        }
}
```

```
//把本类的一个实例添加到js的全局对象window中，
//这样就可以使用window.injs来调用它的方法

webView.addJavascriptInterface(new InJavaScript(), "injs");
```

- 在JS中调用

```
function sendToAndroid(){
        var str = "Cookie call the Android method from js";
        window.injs.runOnAndroidJavaScript(str);//调用android的函数
}
```

# 在Android调用JS函数

- 通过webView的loadUrl方法直接调用

```
Button button = (Button) findViewById(R.id.button);
button.setOnClickListener(new OnClickListener() {
        public void onClick(View arg0) {
                webView.loadUrl("javascript:getFromAndroid(call the js
                        function from Android')");
        }
});
```

- JS方法

```
function getFromAndroid(str){
        alert(str);
}
```

# Android中调试HTML5应用

如果您正在开发一个Android的Web应用程序，您可以使用控制台（console）JavaScript API调试您的JavaScript，调试消息输出到Logcat。

在Android浏览器中用控制台API：

```
Js代码： console.log("Hello World");
Log信息： Console: Hello World http://www.example.com/hello.html :82
```

Android的WebKit没有实现在其他桌面浏览器中实现的所有控制台API。 但您可以使用基本的文本记录功能：

```
console.log (String)
console.info (String)
console.warn (String)
console.error (String)
```

# 在WebView中用控制台API

- 在调试您的WebView的Web页面时，是支持控制台API。在Android 1.6和更低版本下，控制台信息自动发送到Logcat，并加以“WebCore”标签。 如果您是针对Android 2.1（API Level 7）或更高版本，那么你必须提供一个 WebChromeClient 实现 onConsoleMessage() 回调方法，为了使控制台的信息显示到Logcat中。

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.setWebChromeClient(new WebChromeClient() {
    public void onConsoleMessage(String message, int lineNumber, String sourceID) {
        Log.d("MyApplication", message + " -- From line "+ lineNumber + " of "sourceID);
    }
});
```

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.setWebChromeClient(new WebChromeClient() {
    public boolean onConsoleMessage(ConsoleMessage cm) {
        Log.d("MyApplication", cm.message() + " -- From line  “+cm.lineNumber() + " of  “+cm.sourceId() );
        return true;
    }
});
```

ConsoleMessage 还包括一个 MessageLevel 表示控制台传递信息类型。 您可以用messageLevel()查询信息级别，以确定信息的严重程度，然后使用适当的Log方法或采取其他适当的措施。

# 结合PhoneGap进行开发

- 什么是PhoneGap？
- PhoneGap实现原理？
- 如何在Android中集成PhoneGap开发？

# 什么是PhoneGap

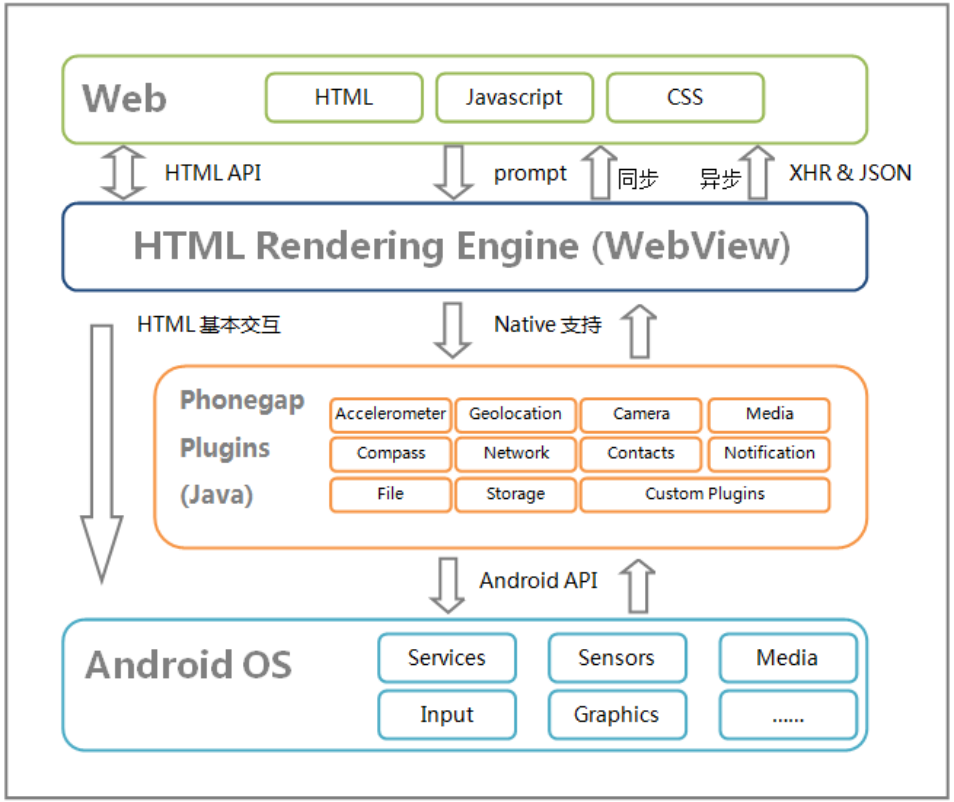
PhoneGap是Nitobi开发的一个免费开源的开发框架，目前最新版本是2.6， 它是一个基于HTML5+CSS和JavaScript的，创建移动跨平台移动应用程序的快速开发平台，以written once, run everywhere一次编译到处运行而受欢迎，只需要改动少量代码而实现跨平台。

PhoneGap 简单来说是一个中间件，为移动前端提供访问移动终端设备及资源的接口。采用统一的标准的html、javascript、css等web技术开发.通过不 同平台的浏览器访问来实现跨平台。通过javascript脚步代码调用系统资源，以降低开发难度。

目前已经对IOS、Android、Bada、BlackBerry、Windows Phone、WebOS、Tizen的支持。

PhoneGap于 2011年10月4日被Adobe宣布收购，后期改名为Cordova，其为PhoneGap奉献给Apache后开源项目。

# PhoneGap实现原理



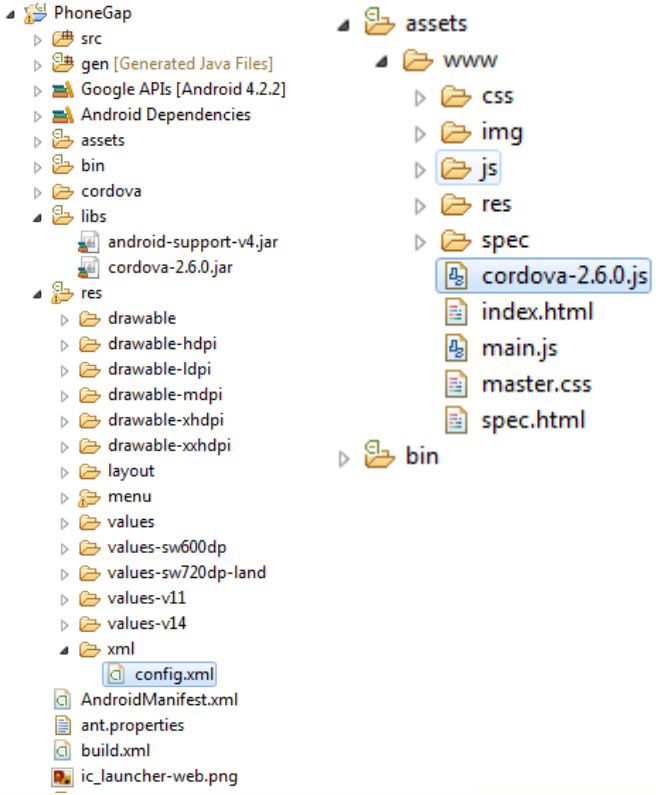
# 如何在Android中集成PhoneGap开发（一）

## 从PhoneGap官网下载并解压Cordova

| Name | Date modified | Type |
|---|---|---|
| android | 2013/4/10 15:27 | File folder |
| bada | 2013/4/10 15:27 | File folder |
| badaWac | 2013/4/10 15:27 | File folder |
| blackberry | 2013/4/10 15:27 | File folder |
| cordova-cli | 2013/4/10 15:27 | File folder |
| ios | 2013/4/10 15:27 | File folder |
| symbian | 2013/4/10 15:27 | File folder |
| webos | 2013/4/10 15:27 | File folder |
| windows8 | 2013/4/10 15:27 | File folder |
| windows-phone-7 | 2013/4/10 15:27 | File folder |
| windows-phone-8 | 2013/4/10 15:27 | File folder |

| | | |
|---|---|---|
| bin | 2013/4/10 15:27 | File folder |
| example | 2013/4/10 15:27 | File folder |
| xml | 2013/4/10 15:27 | File folder |
| changelog | 2013/4/10 15:27 | File |
| cordova-2.6.0.jar | 2013/4/10 15:27 | Executable Jar File |
| cordova-2.6.0.js | 2013/4/10 15:27 | JScript Script File |
| LICENSE | 2013/4/10 15:27 | File |
| NOTICE | 2013/4/10 15:27 | File |
| README.md | 2013/4/10 15:27 | MD File |
| VERSION | 2013/4/10 15:27 | File |

# 如何在Android中集成PhoneGap开发（二）







```
package org.apache.cordova.example;

import android.os.Bundle;

public class cordovaExample extends DroidGap
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        // Set by <content src="index.html" /> in config.xml
        super.loadUrl(Config.getStartUrl());
        //super.loadUrl("file:///android_asset/www/index.html")
    }
}
```

# 如何在Android中集成PhoneGap开发（三）

## 如何调用Native API？

//执行一个通讯录搜索，获取name和email属性
//同时初始化过滤列表到包含“eoe”的联系人记录

```
navigator.contacts.find([‘name’,’email’], success, error, {filter:’eoe’});
function success(contacts){  //获取联系人成功后处理
                for(var i in contacts){
                                alert(contacts[i].name);
                }
}
function error(err){  //获取数据错误时进行处理
                alert(err.code);
}
```

**app/res/xml/config.xml**

```
<plugin name="Contacts" value="org.apache.cordova.ContactManager" />
```

**app/AndroidManifest.xml**

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
```

Home

# 如何在Android中集成PhoneGap开发（四）

如何使用自定义Plugin？

# HTML5 移动开发的特点

## 优点

- 跨平台跨设备
- 易用性
- 提供硬件访问控制
- 可利用成熟javascript框架

## 缺点

- 比原生程序运行慢
- 不适合部分程序
- 内存消耗大
- 调试难度大

# 附录：工具、学习文档

PhoneGap中国：http://www.phonegap.cn/(PhoneGap)
PhoneGap官网：http://www.phonegap.com/ (PhoneGap+Cordova)
Cordova整合文档：http://docs.phonegap.com/en/2.6.0/index.html
jQuery Mobile官网：http://jquerymobile.com/

# Thanks!

www.eoe.cn

最棒的移动开发者社区

关注我们:

@ eoe移动开发者社区