Widget开发指南

-快速掌握Widget应用开发

2012

**中国移动研究院终端技术研究所**

编制历史

|  |  |  |
| --- | --- | --- |
| 版本号 | 更新时间 | 主要内容或重大修改 |
| 1.0.0 | 2012-06-22 | 介绍Widget和Widget开发 |
| 1.0.1 | 2012-06-24 | 修正代码中部分文字描述 |
| 1.1.0 | 2012-06-29 | 增加进阶Widget开发部分内容 |
| 1.2.0 | 2012-07-10 | 调整进阶Widget部分内容 |
| 1.3.0 | 2012-07-20 | 在JS框架部分部分内容 |
| 1.4.0 | 2012-08-20 | 修改部分示例代码 |
| 1.5.0 | 2012-09-13 | 修改部分描述细节 |
| 1.6.0 | 2012-10-29 | 增加单点登录介绍；增加Widget多分辨率适配开发介绍 |
| 1.7.0 | 2012-11-25 | 增加地图能力、支付能力API介绍 |
| 1.8.0 | 2012-11-27 | 整理其他WAC API文档；  增加地图、支付功能示例 |
| 1.9.0 | 2012-12-7 | （由于重复定义）删除地图能力API中Map.reset方法；  （由于重复定义）删除地图能力API中PoiPagedResult.getLocalPage方法；  修改Widget.setPreferenceForKey方法的参数顺序； |

本文档主要起草人：宋培岩

联系方式：[songpeiyan@chinamobile.com](mailto:songpeiyan@chinamobile.com) 15811356851

**目 录**

[**目 录** 4](#_Toc342666883)

[一、 Widget介绍 11](#_Toc342666884)

[二、 Widget应用格式介绍 12](#_Toc342666885)

[1. Widget包结构 12](#_Toc342666886)

[2. 配置文件 13](#_Toc342666887)

[三、 Widget的JavaScript能力扩展API 15](#_Toc342666888)

[四、 开发和调试工具 17](#_Toc342666889)

[五、 入门Widget应用：Hello Widget Example 17](#_Toc342666890)

[1. 创建HTML页面 17](#_Toc342666891)

[2. 创建与HTML页面对应的CSS文件 18](#_Toc342666892)

[3. 创建与HTML页面对应的JS文件 19](#_Toc342666893)

[4. 在PC浏览器上查看效果 19](#_Toc342666894)

[5. 创建Widget应用config文件 20](#_Toc342666895)

[6. 打包Widget应用 21](#_Toc342666896)

[7. 在手机上安装Widget并预览效果 21](#_Toc342666897)

[六、 进阶Widget应用：Message Widget Example 22](#_Toc342666898)

[1. 创建MessageSample.html文件 22](#_Toc342666899)

[2. 使用第三方Web应用开发框架 25](#_Toc342666900)

[3. 使用BAE WebApp开发框架 27](#_Toc342666901)

[4. 使用能力扩展API 29](#_Toc342666902)

[5. 在手机端调试能力扩展API 31](#_Toc342666903)

[七、 Widget多分辨率适配开发 33](#_Toc342666904)

[八、 无线城市Widget应用开发 40](#_Toc342666905)

[1. Widget单点登录功能开发 40](#_Toc342666906)

[2. Widget地图功能开发 43](#_Toc342666907)

[3. Widget支付功能开发 48](#_Toc342666908)

[九、 Q & A 50](#_Toc342666909)

[1. 如何通过JavaScript实现Widget应用退出 50](#_Toc342666910)

[2. 如何解决跨域访问 51](#_Toc342666911)

[十、 附录：资源汇总 52](#_Toc342666912)

[十一、 附录：Feature声明信息 53](#_Toc342666913)

[十二、 附录：Widget能力扩展API文档 54](#_Toc342666914)

[Widget对象 54](#_Toc342666915)

[preferenceForKey() 55](#_Toc342666916)

[setPreferenceForKey() 56](#_Toc342666917)

[openURL() 56](#_Toc342666918)

[onWakeup() 56](#_Toc342666919)

[getDisplayAreaWidth() 57](#_Toc342666920)

[getDisplayAreaHeight() 57](#_Toc342666921)

[getDisplayScale() 57](#_Toc342666922)

[getSSOData() 58](#_Toc342666923)

[getTicket() 59](#_Toc342666924)

[close() 59](#_Toc342666925)

[Device对象 59](#_Toc342666926)

[getAvailableApplications() 60](#_Toc342666927)

[launchApplication() 61](#_Toc342666928)

[vibrate() 61](#_Toc342666929)

[getDeviceId() 61](#_Toc342666930)

[getNetworkConnectionType() 62](#_Toc342666931)

[getOSPlatform() 62](#_Toc342666932)

[getOSVersion() 62](#_Toc342666933)

[ApplicationTypes对象 63](#_Toc342666934)

[ALARM 65](#_Toc342666935)

[BROWSER 65](#_Toc342666936)

[CALCULATOR 65](#_Toc342666937)

[CALENDAR 65](#_Toc342666938)

[CAMERA 65](#_Toc342666939)

[CONTACTS 65](#_Toc342666940)

[FILES 66](#_Toc342666941)

[GAMES 66](#_Toc342666942)

[MAIL 66](#_Toc342666943)

[MEDIAPLAYER 66](#_Toc342666944)

[MESSAGING 66](#_Toc342666945)

[PHONECALL 66](#_Toc342666946)

[PICTURES 67](#_Toc342666947)

[PROG\_MANAGER 67](#_Toc342666948)

[SETTINGS 67](#_Toc342666949)

[TASKS 67](#_Toc342666950)

[Positioninfo对象 67](#_Toc342666951)

[accuracy 68](#_Toc342666952)

[altitude 69](#_Toc342666953)

[altitudeAccuracy 69](#_Toc342666954)

[cellID 69](#_Toc342666955)

[latitude 69](#_Toc342666956)

[longitude 69](#_Toc342666957)

[timeStamp 69](#_Toc342666958)

[DeviceStateInfo对象 70](#_Toc342666959)

[requestPositionInfo() 70](#_Toc342666960)

[onPositionRetrieved() 71](#_Toc342666961)

[AccelerometerInfo对象 71](#_Toc342666962)

[xAxis 72](#_Toc342666963)

[yAxis 73](#_Toc342666964)

[zAxis 73](#_Toc342666965)

[Messaging对象 73](#_Toc342666966)

[createMessage() 74](#_Toc342666967)

[onMessageSendingFailure() 74](#_Toc342666968)

[sendMessage() 74](#_Toc342666969)

[MessageTypes对象 75](#_Toc342666970)

[SMSMessage 75](#_Toc342666971)

[Message对象 76](#_Toc342666972)

[body 77](#_Toc342666973)

[callbacknumber 77](#_Toc342666974)

[destinationAddress 78](#_Toc342666975)

[isRead 78](#_Toc342666976)

[messageId 78](#_Toc342666977)

[messagePriority 78](#_Toc342666978)

[messageType 78](#_Toc342666979)

[sourceAddress 79](#_Toc342666980)

[subject 79](#_Toc342666981)

[time 79](#_Toc342666982)

[validityPeriodHours 79](#_Toc342666983)

[addAddress() 79](#_Toc342666984)

[deleteAddress() 80](#_Toc342666985)

[Multimedia对象 80](#_Toc342666986)

[isAudioPlaying 81](#_Toc342666987)

[getvolume() 81](#_Toc342666988)

[stopAll() 82](#_Toc342666989)

[AudioPlayer对象 82](#_Toc342666990)

[onStateChange() 83](#_Toc342666991)

[open() 84](#_Toc342666992)

[pause() 84](#_Toc342666993)

[play() 84](#_Toc342666994)

[resume() 85](#_Toc342666995)

[Stop() 85](#_Toc342666996)

[Camera对象 85](#_Toc342666997)

[captureImage() 86](#_Toc342666998)

[onCameraCaptured() 86](#_Toc342666999)

[setWindow() 87](#_Toc342667000)

[PIM对象 87](#_Toc342667001)

[createAddressBookItem() 88](#_Toc342667002)

[findAddressBookItems() 88](#_Toc342667003)

[onAddressBookItemsFound() 89](#_Toc342667004)

[getAddressBookItem() 89](#_Toc342667005)

[getAddressBookItemsCount() 90](#_Toc342667006)

[AddressBookItem对象 90](#_Toc342667007)

[address 92](#_Toc342667008)

[addressBookItemId 92](#_Toc342667009)

[company 92](#_Toc342667010)

[eMail 92](#_Toc342667011)

[fullName 92](#_Toc342667012)

[homePhone 93](#_Toc342667013)

[mobilePhone 93](#_Toc342667014)

[title 93](#_Toc342667015)

[workPhone 93](#_Toc342667016)

[getAttributeValue() 93](#_Toc342667017)

[getAvailableAttributes() 94](#_Toc342667018)

[setAttributeValue() 94](#_Toc342667019)

[update() 94](#_Toc342667020)

[CMap对象 95](#_Toc342667021)

[Map对象 95](#_Toc342667022)

[zoom 97](#_Toc342667023)

[type 97](#_Toc342667024)

[center 97](#_Toc342667025)

[Map() 98](#_Toc342667026)

[setCenter() 98](#_Toc342667027)

[setZoom() 98](#_Toc342667028)

[setMapType() 99](#_Toc342667029)

[showTraffic() 99](#_Toc342667030)

[setZoomControlEnabled() 99](#_Toc342667031)

[addOverlay() 100](#_Toc342667032)

[removeOverlay() 100](#_Toc342667033)

[clearOverlays() 100](#_Toc342667034)

[startTapPoint() 101](#_Toc342667035)

[onTapPointGot() 101](#_Toc342667036)

[MapType对象 101](#_Toc342667037)

[MAP\_TYPE\_SATELLITE 102](#_Toc342667038)

[MAP\_TYPE\_BITMAP 102](#_Toc342667039)

[MAP\_TYPE\_VECTOR 102](#_Toc342667040)

[Point对象 102](#_Toc342667041)

[lat 103](#_Toc342667042)

[lng 104](#_Toc342667043)

[Point() 104](#_Toc342667044)

[equals() 104](#_Toc342667045)

[Overlay对象 105](#_Toc342667046)

[Marker对象 105](#_Toc342667047)

[Marker() 106](#_Toc342667048)

[setLabel() 106](#_Toc342667049)

[setBubbleMsg() 107](#_Toc342667050)

[setIcon() 107](#_Toc342667051)

[onBubbleClick() 107](#_Toc342667052)

[closeBubble() 108](#_Toc342667053)

[Circle对象 108](#_Toc342667054)

[Circle() 109](#_Toc342667055)

[setStrokeStyle() 109](#_Toc342667056)

[setStrokeOpacity() 110](#_Toc342667057)

[setFillStyle() 110](#_Toc342667058)

[setFillOpacity() 110](#_Toc342667059)

[setLineWidth() 111](#_Toc342667060)

[Polyline对象 111](#_Toc342667061)

[Polyline() 112](#_Toc342667062)

[setStrokeStyle() 112](#_Toc342667063)

[setStrokeOpacity() 113](#_Toc342667064)

[setLineWidth() 113](#_Toc342667065)

[setPath() 113](#_Toc342667066)

[Polygon对象 114](#_Toc342667067)

[Polygon() 115](#_Toc342667068)

[setStrokeStyle() 115](#_Toc342667069)

[setStrokeOpacity() 115](#_Toc342667070)

[setFillStyle() 116](#_Toc342667071)

[setFillOpacity() 116](#_Toc342667072)

[setLineWidth() 116](#_Toc342667073)

[setPath() 117](#_Toc342667074)

[PoiSearch对象 117](#_Toc342667075)

[poiQuery() 118](#_Toc342667076)

[onPoiSearchComplete() 118](#_Toc342667077)

[PoiPagedResult对象 119](#_Toc342667078)

[getPageCount() 119](#_Toc342667079)

[getPage() 120](#_Toc342667080)

[Position对象 120](#_Toc342667081)

[getName() 121](#_Toc342667082)

[getPoint() 121](#_Toc342667083)

[getAddress() 121](#_Toc342667084)

[getPostcode() 122](#_Toc342667085)

[getPhone() 122](#_Toc342667086)

[RouteSearch对象 122](#_Toc342667087)

[TransitMode 123](#_Toc342667088)

[DrivingMode 124](#_Toc342667089)

[routeQuery() 124](#_Toc342667090)

[onRouteSearchComplete() 124](#_Toc342667091)

[RouteResult对象 125](#_Toc342667092)

[getRouteResultSize() 125](#_Toc342667093)

[getRoute() 126](#_Toc342667094)

[Route对象 126](#_Toc342667095)

[getStartPoint() 127](#_Toc342667096)

[getEndPoint() 127](#_Toc342667097)

[getSegments() 128](#_Toc342667098)

[getDistance() 128](#_Toc342667099)

[show() 128](#_Toc342667100)

[removeFromMap() 129](#_Toc342667101)

[Segment对象 129](#_Toc342667102)

[getConsumeTime() 131](#_Toc342667103)

[getFirstPoint() 131](#_Toc342667104)

[getLastPoint() 131](#_Toc342667105)

[getLength() 132](#_Toc342667106)

[getShapes() 132](#_Toc342667107)

[getFirstStationName() 132](#_Toc342667108)

[getLastStationName() 133](#_Toc342667109)

[getLineName() 133](#_Toc342667110)

[getOffStationName() 133](#_Toc342667111)

[getPassStops() 134](#_Toc342667112)

[getStopNumber() 134](#_Toc342667113)

[getActionDescription() 134](#_Toc342667114)

[getRoadName() 135](#_Toc342667115)

[Location对象 135](#_Toc342667116)

[requestLocation() 136](#_Toc342667117)

[onMyLocationComplete() 136](#_Toc342667118)

[CPay对象 136](#_Toc342667119)

[urlPay() 137](#_Toc342667120)

[onUrlPayFinished() 137](#_Toc342667121)

1. Widget介绍

Widget是运行在手机上采用HTML/CSS/JavaScript编写的Web应用程序。在开发Widget应用时，可以使用传统Web开发中的所有功能和API接口，除此之外，Widget的运行环境中还扩展了很多移动终端及业务能力接口，通过这些接口可以提供通讯录、多媒体、终端信息等终端能力及位置、地图等业务能力，这些接口的调用方法与传统JavaScript API接口的调用方法基本相同，不会增加使用门槛。由于这种Widget应用的应用格式遵循WAC规范，所以我们就称这些应用为WAC Widget，一般简称为Widget应用。

Widget是需要运行在自己特定的运行环境上，这个运行环境被称做**Widget引擎**，为了便于理解，你可以把Widget引擎看成一个扩展版的浏览器，用HTML/CSS/JavaScript等Web技术开发的Widget应用就像一个打包后的网页，可以在上面运行。当然Widget应用虽然是使用Web技术开发的，但是它与我们传统应用也是一样的，在使用时需要先安装，如果不需要了，可以卸载，因此，在Widget引擎中除了提供Widget应用的运行环境，还包含了对Widget应用的安装卸载管理。

为了便于理解Widget应用、Widget引擎和操作系统的关系可以参看下图：

将系统的Native API转换为Web API

Web内容的解析展示能力

Widget应用的

安装和卸载

**Widget管理器**

**Widget运行环境**

Widget应用

Widget应用

Widget应用

JavaScript核心

Web核心

**浏览器内核**（例如：webkit等）

Telephony

**移动终端系统API**

Message

…

Device

**OS** (Android, iOS, WindowsPhone…)

**Widget**

**引 擎**

Widget应用

1. Widget应用格式介绍
2. Widget包结构

每个Widget应用包都是一个符合ZIP规范的压缩文件，可以通过ZIP解压缩工具解压，在压缩包根目录下可以看到包含以下文件： config.xml配置文件、icon图标文件、HTML、CSS、JS以及图片等资源文件。如下图所示：

HelloWidgetApp.wgt

index.html

HelloWidget.js

images

HelloWidget.css

config.xml

icon.png

HelloWidget.jpg

Logo.jpg

每个Widget程序必须存在至少一个HTML文件（例如：index.html）作为Widget的入口，并通过config.xml配置文件中的content标签进行定义。

1. 配置文件

Widget的配置文件是Widget包根目录下的名为“config.xml”(注：文件名为小写)的XML文档，含有Widget的属性和配置信息，是Widget的必需文件。

一个典型的config.xml代码如下：

<?xml version="1.0" encoding="UTF-8" ?>

<widget id="com.cmcc.HelloWidget"

version="01.00.Beta"

versioncode="1"

height="200"

width="300">

<name>HelloWidget</name>

<description>An example Widget</description>

<icon src="icon.png"/>

<license href="http://www.apache.org/licenses/LICENSE-2.0.html">Apache v2</license>

<author email="Widget@chinamobile.com">Widget</author>

<content src="index.html" />

<feature name="http://jil.org/jil/api/1.1/widget"/>

<feature name="http://jil.org/jil/api/1.1/messaging"/>

<feature name="cmap"/>

<needwcitySSO>false</needwcitySSO>

</widget>

config.xml中包含的字段含义如表2-1所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段含义 | | | |
| Widget | ID \* | Widget通用标识 | | |
| Version | Widget版本，系统默认为01.00beta | | |
| VersionCode | Widget版本编号，数据类型为整数，系统默认为1 | | |
| Height | Widget默认的屏幕显示高度 | | |
| Width | Widget默认的屏幕显示宽度 | | |
| Name | Widget名字 | | | |
| Description | Widget文字描述信息 | | | |
| Icon \* | Widget图标 | | | |
| License | 许可证信息 | | | |
| Name | | | 许可证名称 |
| Link Reference | | | 许可证引用链接 |
| Author | Widget作者信息 | | | |
| Name | | | 姓名 |
| E-Mail | | | 邮件地址 |
| Content \* | 规定的程序呈现入口，将显示的第一个页面 | | | |
| Feature | Feature声明信息，声明使用哪些扩展API  每个Feature对应的声明信息可参考附录1。 | | | |
| Name | | API声明 | |
| needwcitySSO | 是否有单点登录功能（仅限无线城市业务使用） | | | |

\*标注的项为必选项，其他为可选项。

1. Widget的JavaScript能力扩展API

在Web开发中，JavaScript所使用的API主要都是用来操作DOM节点和节点的CSS样式，没有合适的API来调用手机终端和业务能力，例如：操作短信、获取联系人信息、获取位置、地图显示等。为了弥补这个不足，在Widget引擎中扩展了很多JavaScript API，以实现Widget应用访问手机终端和业务能力的需要，我们称之为JavaScript能力扩展API。能力扩展API可以分为终端能力扩展API和业务能力扩展API。

终端能力扩展API包括：

1. PIM类，包含与个人信息管理相关的API，例如：通信录等。
2. Messaging类，包含与消息相关的API，例如：短/彩信等。
3. Multimedia类，包含与多媒体相关的API，例如：相机、音频等。
4. Device类，包含与设备相关的API，例如：设备信息、位置信息等。

业务能力扩展API目前包括：

1. CMap类，包含与定位、地图相关的API。
2. CPay类，包含与支付相关的API。

业务能力扩展API将会在后续持续增加。

能力扩展API由于涉及到终端和业务能力调用，基于安全考虑，在使用这些API时需要在Widget应用配置文件config.xml中通过<feature>标签声明所要使用的API类型，否则，将不能使用。每个Feature对应的声明信息可参考**“附录：Feature声明信息”**。假设在Widget中使用到了Widget类、Messaging类和CMap类的API，则需要在config.xml中以如下方式声明：

<feature name="http://jil.org/jil/api/1.1/widget"/>

<feature name="http://jil.org/jil/api/1.1/messaging"/>

<feature name="cmap"/>

在config.xml中声明过后，就可以在Widget应用的网页中通过JavaScript调用对应API。能力扩展API的调用非常简单，与JavaScript中调用普通的API完全相同。具体实例介绍如下：

Messaging类中包含了管理Message相关的函数，下面以发送一条短消息为例，介绍如何调用创建和发送Message的API，示例如下：

//创建一条短信

var msg = Widget.Messaging.createMessage(Widget.MessageTypes.SMSMessage);

msg.addAddress("destination"，"13900001111");

msg.body="Test send SMS";

//指定发送失败的回调处理函数

Widget.Messaging.onMessageSendingFailure=function myCallBack(msg, error) {

//错误处理

}

//发送短信

Widget.Messaging.sendMessage(msg);

上面简单介绍了能力扩展API的使用，在开发Widget过程中，如果需要查询详细介绍和使用方法可以参考**“附录：Widget能力扩展API文档”**。

1. 开发和调试工具

虽然Widget是以独立应用的方式运行在手机上，但Widget仍是采用HTML、CSS、JavaScript编写的Web应用程序，因此其开发工具与传统的Web开发是一样的。可以根据个人的喜好选择最适合自己工具。

1. 入门Widget应用：Hello Widget Example

前几节已经介绍了Widget相关的基础知识，现在一切准备工作已经就绪，我们来一起开发第一个Widget应用。通过这个Widget应用的例子，可以学习到开发一个Widget的完整的步骤，在平时的开发中我们也是按照这些步骤最终完成Widget的开发。

1. 创建HTML页面

创建一个HelloWidget.html文件，文件源代码如下：

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<title>Hello Widget</title>

<link href="HelloWidget.css" rel="stylesheet" type="text/css"/>

<script type="text/JavaScript" src="HelloWidget.js"></script>

</head>

<body>

<div id="front" onclick="changeText()">

<div id="hello">Hello Widget</div>

</div>

</body>

</html>

从代码中可以看出与HTML链接的CSS样式文件为HelloWidget.css, 与HTML链接的JavaScript文件为HelloWidget.js。这两个文件的内容在后面将会介绍。<body>中包含了两个Div，id为“front”的Div中还包含了一个id为“hello”的标签，后者区域内的文字内容为“Hello Widget”；同时Div “front”上通过onClick绑定了点击事件，当该Div被点击时，将会调用changeText()函数，该函数被定义在HelloWidget.js中。

1. 创建与HTML页面对应的CSS文件

创建一个HelloWidget.css文件，文件源代码如下：

#front {

position: absolute;

background-image: url("Default.png");

top: 0px;

left: 0px;

width: 272px;

height: 159px;

}

#hello {

position: absolute;

color:white;

top:65px;

left:65px;

width:180px;

height: 50px;

font:24px "verdana";

}

以上两个CSS语句块规定了Html中id为“front”和“hello”的Div的呈现样式，包括：背景图片，位置，字体颜色，字体大小等。

1. 创建与HTML页面对应的JS文件

创建一个HelloWidget.js文件，文件源代码如下：

function changeText(){

if(document.getElementById("hello").innerHTML == "Hello Widget！")

document.getElementById("hello").innerHTML="I Love Widget!";

else

document.getElementById("hello").innerHTML="Hello Widget！";

}

该段JavaScript代码定义了一个函数changeText()，该函数首先检测id为“hello”的元素内容是否为“Hello Widget！”，如果是则将文字变为“I Love Widget！”，否则重新写为“Hello Widget！”。

1. 在PC浏览器上查看效果

在PC浏览器中打开HelloWidget.html查看效果，点击灰色背景区域，文字内容将在“Hello Widget！”和“I Love Widget！”之间切换。如果显示有问题或JavaScript的行为执行不正确，可以使用Web开发相关的调试工具跟踪问题的原因，然后修改代码，重新查看效果，直到正常运行。

1. 创建Widget应用config文件

Web开发部分已经完成，接下来我们要编写Widget应用的配置文件。

创建一个config.xml文件，内容如下：

<?xml version="1.0" encoding="UTF-8" ?>

<widget id="com.cmcc.HelloWidget"

version="01.00.Beta"

versioncode="1"

height="200"

width="300">

<name>HelloWidget</name>

<description>An example Widget</description>

<author email="Widget@chinamobile.com">Widget</author>

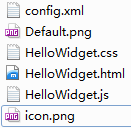
<icon src="icon.png"/>

<content src="HelloWidget.html" encoding="utf-8"/>

</widget>

该配置文件中，特别需要注意的有三个部分：（1）id是Widget的唯一标识，不能与其他Widget重复。（2）icon标签指定了Widget的图标，一定要保证图标资源文件的名称和大小写正确。（3）content标签指定了Widget启动时对应打开的html页面，一定要保证该html文件的名称和大小写正确。

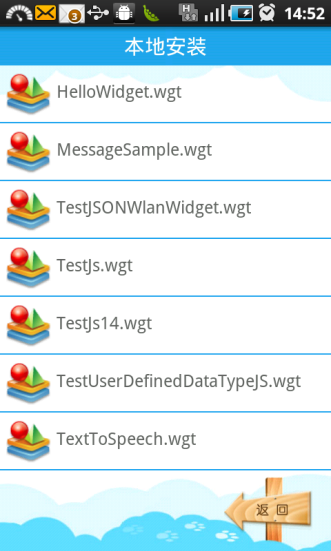
1. 打包Widget应用



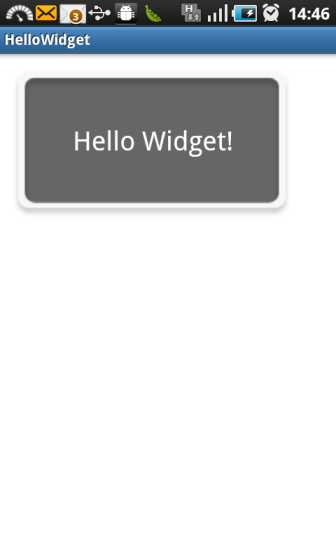
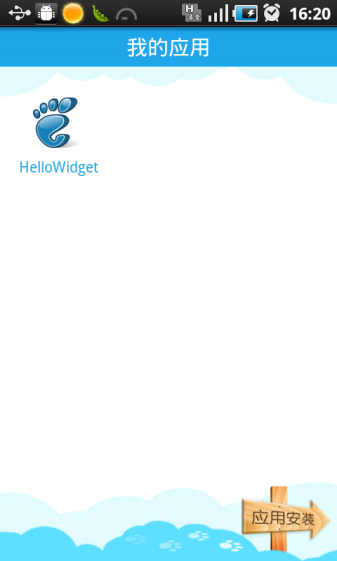
将我们创建的文件和图片资源文件用压缩工具打包为HelloWidget.zip文件，然后将.zip后缀修改为.wgt，现在我们已经完成了第一个Widget应用程序：HelloWidget.wgt（[点击下载](http://s2.cmribrowser.com/WebAppLib/HelloWidget.wgt)）。

1. 在手机上安装Widget并预览效果

以Android平台为例，按以下步骤在手机上预览效果，（1）检查手机上是否安装了“Widget运行客户端”（[点击下载](http://s2.cmribrowser.com/WebAppLib/WidgetClient_V1.2.apk)），如果没有，请先下载安装。（2）将HelloWidget.wgt拷贝到手机的SD卡上。（3）从“Widget运行客户端”中安装HelloWidget.wgt到手机。界面如下：



（4）查看运行效果



1. 进阶Widget应用：Message Widget Example

上一节中介绍了开发一个Widget的具体步骤，按照这些步骤可以完成简单Widget的开发。在本节中将会介绍一些更高阶的Widget开发内容，下面将结合一个可以读取收件箱短信的Widget进行介绍。

1. 创建MessageSample.html文件

创建MessageSample.html文件，源代码如下：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>MessageSample</title>

<link href="jquery-mobile/jquery.mobile-1.0.css" rel="stylesheet" type="text/css" />

<script src="MessageSample.js" type="text/JavaScript"></script>

<script src="jquery-mobile/jquery-1.7.1.js" type="text/JavaScript"></script>

<script src="jquery-mobile/jquery.mobile-1.0.js" type="text/JavaScript"></script>

<style type="text/css">

.pop\_box {

z-index: 9999; /\*这个数值要足够大，才能够显示在最上层\*/

top:75%;

display: none;

position: absolute;

width:100%;

height:24px;

vertical-align:central;

}

</style>

//实现Toast效果功能

<script>

$(document).ready(function(){

$("button").click(function(){

prompt.init(content);

});

});

var prompt=(function(){

return{

"init":function(con){

var toast = $("#toast\_view");

var toast\_content = "正在读取第" + getPoisiton() +"条短信内容";

$("#toast\_content").html(toast\_content);

toast.fadeIn(400);

setTimeout(function(){

toast.fadeOut(400);

},1500);

}

}

})();

</script>

</head>

<body>

<div data-role="page" id="MainPage" data-theme="d" style="position:relative;">

<div data-role="footer" data-theme="b">

<h5>Message Sample</h5>

</div>

<div data-role="content" data-theme="d">

<div data-role="fieldcontain" id="acount\_input" style="height:302px">

<div align="center" data-role="controlgroup" data-type="horizontal">

<Button onclick="onGetPreviousSms()" data-theme="b"><<</Button>

<Button onclick="onGetCurrentSms()" data-theme="b" >最新短信</Button>

<Button onclick="onGetNextSms()" data-theme="b">>></Button>

</div>

<textarea name="textinput" id="textinput" style="height:150px">显示短信内容...</textarea>

<p>

</div>

</div>

<!-显示toast的div->

<div id="toast\_view" class = "pop\_box ui-body ui-body-e" align="center">

<fieldset class="ui-grid-a">

<div class="ui-block-b" align="center" id="toast\_content" style="text-align:center;width:100%"></div>

</fieldset></div>

<div data-role="footer" id="bottom\_view" data-theme="b" style="position:absolute; left:0; bottom:0;">

<h5>中国移动研究院版权所有<br /></h5>

</div>

</div>

</body>

</html>

页面中主要包含三个Button和一个Textarea，三个Button都绑定了事件响应，分别用于读取上一条短信，最新短信和下一条短信，Textarea用于显示读取出来的短信内容。 MessageSample.js是自己编写的JavaScript文件，这个文件的内容将在后面介绍。

1. 使用第三方Web应用开发框架

MessageSample.html中的以下代码是为了引用第三方Web应用框架：

<link href="jquery-mobile/**jquery.mobile-1.0.css**" rel="stylesheet" type="text/css" />

<script src="jquery-mobile/**jquery.mobile-1.0.js**" type="text/JavaScript"></script>

<script src="jquery-mobile/**jquery-1.7.1.js**" type="text/JavaScript"></script>

其中jquery-1.7.1.js对应的是目前非常流行的免费开源JavaScript框架jQuery，该框架比较突出的优势在于：（1）强大的选择器， 通过这些选择器可以用最少的代码做最多的事，提高开发效率。（2）JS脚本与HTML页面内容分离，使程序结构更清晰。（3）支持扩展插件，通过借助已经存在的丰富插件，节省开发时间。

jquery.mobile-1.0.css和jquery.mobile-1.0.js对应的是免费开源Web应用框架jQueryMoblie。这个框架是包含界面的，在使用时必须依赖于jQuery框架。使用第三方UI框架的优势在于可以快速的开发出漂亮的UI界面，而且框架对平台的兼容性进行了处理，所以几乎不需要考虑分辨率等兼容性问题，这样可以极大提高开发时间。同样适用第三方框架的缺点也很明显，UI框架中的界面样式都是设定好的，只能从提供的几套样式中选择。总体来说，如果对界面有特定的要求，例如：自己设计了一套UI资源，Widget中的界面都要按照这套UI资源实现，在这种情况下，不建议使用jQuery Moblie。否则，如果对界面样式没有特别要求，则可以使用jQuery Moblie来减少界面开发的工作量。

用PC浏览器查看使用了jQuery Moblie的MessageSample.html页面显示效果如下：



1. 使用BAE WebApp开发框架

为了在该Widget应用上体现更好的用户体验，我们想要在点击读取短信Button时，弹出一个提示，显示几秒后自动消失，告知用户正在读取第几条短信息（即类似Android手机上的Toast效果）。

要实现这样的效果，可以借助“[BAE WebApp开发框架](http://s2.cmribrowser.com/WebAppLib/index.html)（请使用Safari或Chrome打开）”，该框架在jQuery等开源框架基础上实现了多种最常用的Web应用特效控件和动画效果。根据该框架中提供的示例程序代码，可以很方便的将这些特效控件和动画效果加入到自己的应用中。

为了实现Toast效果，可以使用该框架中的消息提示控件。在MessageSample.html中对应的代码如下：

//实现Toast效果功能

<script>

$(document).ready(function(){

$("button").click(function(){

prompt.init(content);

});

});

var prompt=(function(){

return{

"init":function(con){

var toast = $("#toast\_view");

var toast\_content = "正在读取第" + getPoisiton() +"条短信内容";

$("#toast\_content").html(toast\_content);

toast.fadeIn(400);

setTimeout(function(){

toast.fadeOut(400);

},1500);

}

}

})();

</script>

首先为三个Button绑定click事件，当任何一个Button被点击时将调用Prompt.init()函数。然后在该init()函数中，为Toast设定文字内容并显示Toast，同时设定一个定时器在1.5s后将隐藏Toast。

MessageSample.html中的以下代码就是Toast对应的Div，在该Div上显示Toast的内容。

<!-显示toast的div->

<div id="toast\_view" class = "pop\_box ui-body ui-body-e" align="center">

<fieldset class="ui-grid-a">

<div class="ui-block-b" align="center" id="toast\_content"

style="text-align:center;width:100%"></div>

</fieldset></div>

效果如下：



1. 使用能力扩展API

创建MessageSample.js文件，源文件代码如下：

var position=0;

function onGetCurrentSms() {

position = 0;

var msg = Widget.Messaging.getMessage(

Widget.Messaging.MessageTypes.SMSMessage,

"inbox",

position);

$("#textinput")[0].value=msg.body;

Widget.Device.vibrate(1);

}

function onGetPreviousSms() {

if(--position <= 0) {

position=0;

}

var msg = Widget.Messaging.getMessage(

Widget.Messaging.MessageTypes.SMSMessage,

"inbox",

position);

$("#textinput")[0].value=msg.body;

Widget.Device.vibrate(1);

}

function onGetNextSms() {

if(++position <= 0) {

position=0;

}

var msg = Widget.Messaging.getMessage(

Widget.Messaging.MessageTypes.SMSMessage,

"inbox",

position);

$("#textinput")[0].value=msg.body;

Widget.Device.vibrate(1);

}

function getPoisiton(){

return position + 1;

}

该JS文件中，包含三个函数onGetCurrentSms()、onGetPreviousSms()和onGetNextSms()分别表示获取第一条，前一条，后一条短消息，使用到能力扩展API中的Messaging类中的getMessage()方法。

var msg =

Widget.Messaging.getMessage(

//消息类型为短消息

Widget.Messaging.MessageTypes.SMSMessage,

//从收件箱中读取

"inbox",

//由Position的值指定读取第几条短消息

position);

为了在点击读取短信按钮时给用户反馈，调用了Device类中的震动函数vibrate()。

//参数1代表震动持续时间，单位为秒

Widget.Device.vibrate(1);

1. 在手机端调试能力扩展API

由于只有Widget引擎支持能力扩展API，因此如果要验证Widget中调用能力扩展API的逻辑是否正确，只能将Widget运行在Widget引擎上进行验证。根据上一节的介绍，将网页打包成wgt格式需要有config.xml, 因此接下来我们需要为Widget编写config.xml文件，源代码如下：

<?xml version="1.0" encoding="UTF-8" ?>

<widget id="com.cmcc.MessageSample"

version="01.00.Beta"

versioncode="1"

height="800"

width="480">

<name>Message Sample</name>

<description>An example widget</description>

<author email="widget@chinamobile.com">widget</author>

<icon src="icon.png"/>

<content src="MessageSample.html" encoding="utf-8"/>

<feature name="http://jil.org/jil/api/1.1/messagetypes"/>

<feature name="http://jil.org/jil/api/1.1/messaging"/>

<feature name="http://jil.org/jil/api/1.1/message"/>

<feature name="http://jil.org/jil/api/1.1/device"/>

</widget>

按照上节中所介绍的知识，设置配置文件中id、icon和content标签对应的值。由于在本Widget中调用了能力扩展API，因此需要在配置文件中通过<feature>标签声明所使用到的API类。

<feature name="http://jil.org/jil/api/1.1/messagetypes"/>

<feature name="http://jil.org/jil/api/1.1/messaging"/>

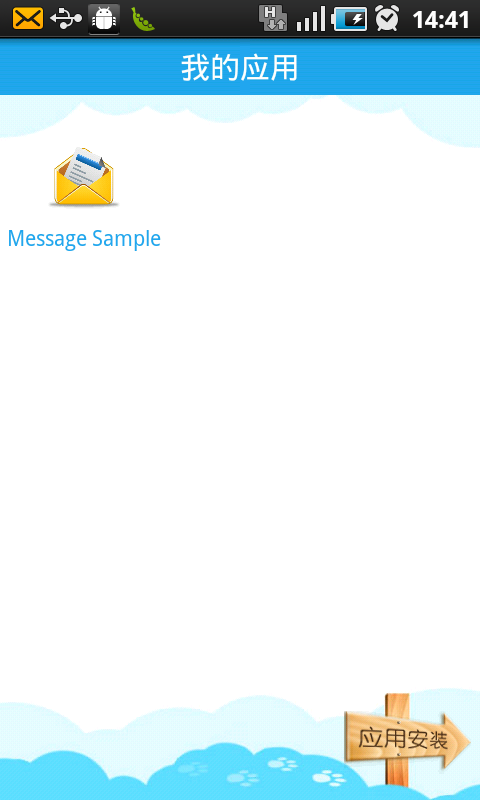
<feature name="http://jil.org/jil/api/1.1/message"/>

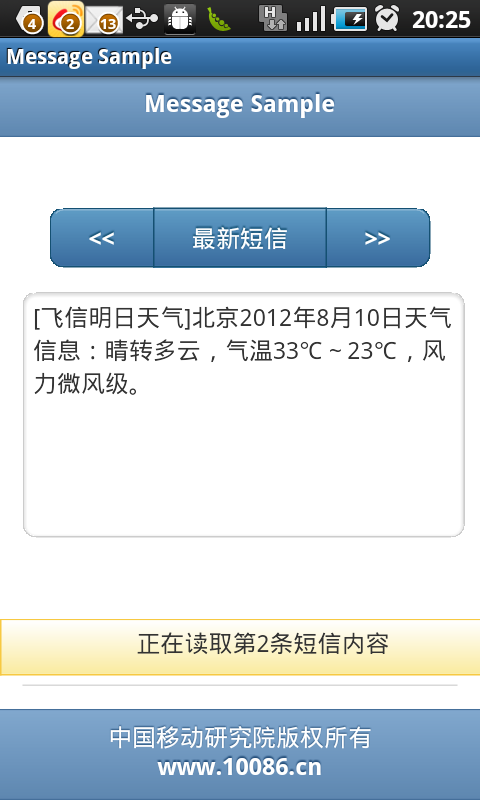
<feature name="http://jil.org/jil/api/1.1/device"/>

从MessageSample.js中可以找到使用MessageTypes、Messaging和Device类的代码，很明显这四个类需要声明。但是Message类在代码中没有找到为什么还需要声明呢？是因为Widget.Messaging.getMessage()函数的返回值是一个Message对象，因此Message类也需要声明。

将网页内容打包为wgt格式，并安装到Widget运行客户端上查看效果。点击网页上的“最新短信”按钮，以及“《”和“》”按钮，查看能否正确显示短信内容，并伴随手机震动。如果工作不正常，需要在网页中增加Log信息，然后重复**打包**->**安装到手机**->**查看效果**三个步骤，跟踪问题原因，直到解决。

最终完成以后，MessageSample.wgt（[点击下载](http://s2.cmribrowser.com/WebAppLib/MessageSample.wgt)）的效果如下：

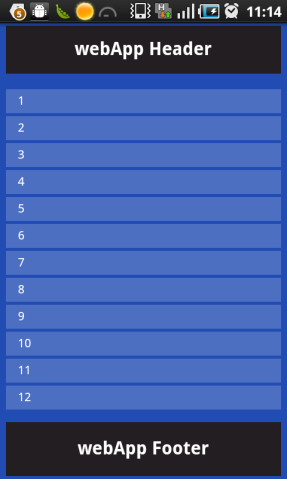
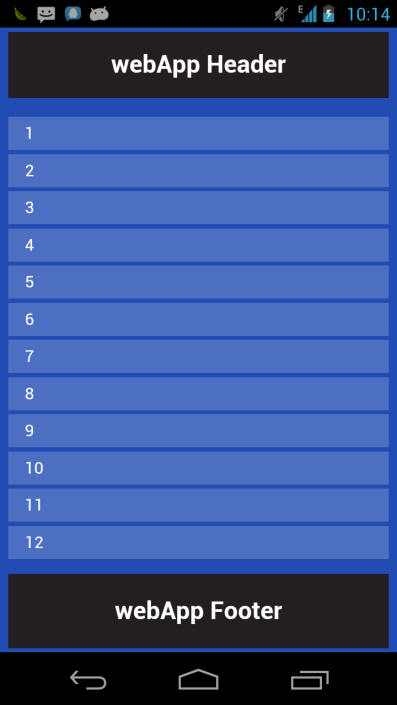
 

1. Widget多分辨率适配开发

可通过Widget引擎缩放Widget页面使其匹配不同屏幕分辨率。Widget的config.xml配置文件中有width属性，该属性代表Widget的默认宽度，如果该值大于0并且与屏幕实际宽度值不同，Widget引擎会自动缩放Widget页面大小，从而使得Widget的宽度刚好与屏幕宽度相适应。假设Widget页面大小为480\*320，手机屏幕大小为800\*480，Widget引擎将会自动将Widget页面放大1.5倍，放大后的Widget宽度为320\*1.5=480刚好与屏幕宽度相同。

由于不同分辨率屏幕的高宽比不一定相同，因此Widget缩放之后有可能会造成宽度已经与屏幕适配但高度仍然不合适。在上面的例子中， Widget页面放大1.5倍后，由于是等比例缩放，高宽比不变，Widget高度也就变为480\*1.5=720，而屏幕高度为800，二者高度并不相同。为了解决这种问题，**在开发Widget时，开发者需要将Widget页面设计成可以根据屏幕高度自动调整布局**，在Web开发中也经常会这样设计。

下面以ScreenAdapt\_1.wgt（[点击下载](http://s2.cmribrowser.com/WebAppLib/ScreenAdapt_1.wgt)）为例，介绍具体的实现方法。在不同分辨率下ScreenAdapt\_1 Widget的显示效果如下：

480\*320 800\*480 1280\*720

样例代码如下：

<!doctype html>

<html>

<head>

<meta charset="utf-8"/>

<meta align="center" name="viewport" content="user-scalable=1"/>

<style>

body, header, footer, section, div, ul, li {

margin: 0;

padding: 0;

}

body {

background: #214CB4;

}

ul {

list-style: none

}

//网页中使用了绝对宽度，是按照480px为宽度设计的页面。

.ui-header, .ui-section, .ui-footer {

width: 460px;

position: absolute;

left: 10px;

overflow: hidden;

}

.ui-header, .ui-footer {

z-index: 50;

background: #231E21;

font-size: 30px;

font-weight: bold;

color: white;

text-align: center;

}

.ui-header {

height: 80px;

top: 5px;

line-height: 80px;

}

.ui-footer {

height: 90px;

bottom: 5px;

line-height: 90px;

}

.ui-section {

/\* top : 100px; \*/

}

.ui-list {

width: 100%;

}

.ui-list li {

margin-bottom: 5px;

width: 100%;

height: 40px;

background: rgba(255, 255, 255, .2);

color: white;

line-height: 40px;

text-indent: 20px;

font-size: 20px;

}

</style>

</head>

<body>

<div class="appScreenAdapt" id="appScreenAdapt">

<header class="ui-header">webApp Header</header>

<section class="ui-section" id="section">

<ul class="ui-list">

<li>1</li>

<li>2</li>

<li>3</li>

<li>4</li>

<li>5</li>

<li>6</li>

<li>7</li>

<li>8</li>

<li>9</li>

<li>10</li>

<li>11</li>

<li>12</li>

<li>13</li>

<li>14</li>

<li>15</li>

<li>16</li>

<li>17</li>

<li>18</li>

<li>19</li>

</ul>

</section>

<footer class="ui-footer">webApp Footer</footer>

</div>

<script>

//获取显示网页区域的宽度和高度，单位为像素。

var displayWidth = **Widget.getDisplayAreaWidth();**

var displayHeight = **Widget.getDisplayAreaHeight();**

//获取当前页面的缩放比例

var displayScale = **Widget.getDisplayScale();**

//由于页面进行了缩放，需要计算出没有缩放时对应的高度。

var HeightDefaultScale = displayHeight / displayScale;

//根据屏幕尺寸设定应用的高宽,自适应应用的高度,与应用居中

(function (screenHeight) {

/\*

\* 根据可视区域高度screenHeight计算出可显示条数,以及上下空白区的高度

\*/

var

appScreenAdapt,

section,

sectionHeight = screenHeight - (80 + 5)/\*header height + space height\*/ - (90 + 5)/\*footer height + space height\*/ - (5 + 5)/\*above space default height + below space default height\*/,

space = sectionHeight % (40 + 5)/\*eache section height + space height\*/,

top = 85;

/\*first section position\*/

sectionHeight = sectionHeight - space;

top = (space + 10) / 2 + 5 + top;

appScreenAdapt = document.getElementById('appScreenAdapt');

section = document.getElementById('section');

appScreenAdapt.setAttribute('style', 'display:block; height:' + screenHeight + 'px; position:relative');

section.setAttribute('style', 'display:block; height:' + sectionHeight + 'px; top:' + top + 'px');

})(HeightDefaultScale);

</script>

</body>

</html>

该页面的宽度在CSS中已经设定为480px，为了让Widget引擎自动缩放该页面从而自适应不同分辨率，需要在Widget的config.xml中设定width=480。

<widget xmlns="http://www.jil.org/ns/widgets"

id="6e18fb5f-cddf-438f-bce1-eafdfa6ad49b/ScreenAdapt\_1"

version="01.00.Beta" versioncode="1"

height="200" width="480">

为了让WebApp Footer在不同分辨率下始终位于页面最下方，需要采用相对布局。具体方法如下：

1. 在appScreenAdapt中加入position:relative样式，使得他的子元素以appScreenAdapt为参考设定相对位置。
2. 在ui-footer中增加Position:absolute； bottom:5px；样式，让ui-footer位于距离appScreenAdapt Div的最底部5px的位置。

界面中的ui-section高度是不确定的，会随着不同分辨率而改变，这部分内容的布局需要根据具体的高度灵活改变。从UI效果图可以看出，在不同分辨率下，显示条数会有不同。通过Widget引擎提供的接口，可以获取到在不同分辨率下网页显示区域的宽度、高度以及缩放比例，代码如下：

//获取显示网页区域的宽度和高度，单位为像素。

var displayWidth = **Widget.getDisplayAreaWidth();**

var displayHeight = **Widget.getDisplayAreaHeight();**

//获取当前页面的缩放比例

var displayScale = **Widget.getDisplayScale();**

//由于页面进行了缩放，需要计算出没有缩放时对应的高度。

var HeightDefaultScale = displayHeight / displayScale;

根据这些信息后，就可以计算出ui-section在不同屏幕分辨率下的实际高度，然后根据这个高度值对该区域进行布局。通过这样的方法ui-section的布局是根据不同分辨率灵活改变的，因此可以适应不同的分辨率。

**注：网页的meta中不能指定width和initial-scale，否则上述自适应方法都可能会失效。**

<meta align="center" name="viewport" content="~~width=device-width, initial-scale=1~~" />

1. 无线城市Widget应用开发
2. Widget单点登录功能开发

单点登录（Single Sign On）是针对多个应用，用户只需要登录一次就可以访问所有相互信任的应用。

对于无线城市客户端的单点登录，客户端在登录后可以获得用户信息并可以生成T-Ticket身份凭证，Widget应用在需要单点登录时，可以在config.xml文件中做出声明，并通过Widget引擎提供的单点登录JavaScript API获取用户信息，或者获取T-Ticket身份凭证。

**注：Widget引擎提供的单点登录API仅能在无线城市客户端上使用。**

**a) Widget应用单点登录流程**

如果Widget应用只需获取用户信息，不需登录应用服务器平台进行身份验证

如果Widget应用需要登录应用服务器平台进行身份验证

开始

用户客户端登录

客户端存储ticket和用户信息

匿名用户

登录用户

登录应用服务器平台, 请求中携带t-ticket

结束

N

N

Y

Y

启动Widget应用

通过getSSOData()方法Widget获取用户信息成功？

通过getTicket()方法Widget获取T-Ticket身份凭证成功？

**b) 使用单点登录示例**

若Widget应用在需要单点登录，首先要在config.xml文件中做出feature声明并设置<needwcitySSO>标签，示例如下:

<feature name="singlesign"/>

<needwcitySSO>true</ needwcitySSO>

之后在JavaScript代码中，可使用两种方法来进行单点登录：

**方法一**：适用于Widget应用只需获取用户信息，不需跟应用服务器进行用户鉴权认证的情况。

使用getSSOData函数获取用户信息。比如获取用户信息“name”的JavaScript代码如下：

var username = **Widget.getSSOData**("name");

**方法二**：适用于Widget应用需要跟应用服务器进行用户鉴权认证的情况。

使用getTicket函数获取T-Ticket身份凭证，在向应用服务器平台发起身份验证的请求中以地址参数形式带上T-Ticket身份凭证。如原始请求地址[http://www.xxx.com](http://www.xxxx.com)，带上T-Ticket后为[http://www.xxx.com](http://www.xxxx.com)?t\_ticket=myTicket。下面以使用jQuery框架的JavaScript代码片段为例，发送一个带T-Ticket身份凭证的登录请求：

var myTicket = **Widget.getTicket();**

$.ajax( {

type: "GET",

url: requestUrl,

cache: false,

//以参数形式带上Ticket身份凭证

data: {t\_ticket: myTicket },

dataType: "json",

timeout: 25000,

success: function (data) { … },

error: function(){ … }

});

应用服务器收到应用发送过来的T-Ticket凭证后，继续向无线城市省平台进行身份验证鉴权，完成后续流程。

1. Widget地图功能开发

Widget引擎提供了地图能力扩展API，供Widget应用调用，从而实现地图功能定制。地图功能API请参考附录API文档——CMap对象。

**a) 使用地图能力示例**

若Widget应用需要使用地图能力，首先要在config.xml文件中做出feature声明，如下:

<feature name="cmap"/>

使用CMap API在Widget应用中显示一张地图相当简单。首先我们需要在DOM中定义一个地图显示的区域节点，通常使用DIV，然后在JavaScript代码中使用节点的ID来构造一个map对象，就可以显示一张默认的地图了。例如：

<html xmlns="http://www.w3.org/1999/html">

<head>

<title>位置/地图能力使用示例</title>

<meta charset="utf-8">

<style type="text/css">

\* {

font-size: 18pt;

}

body {

width:100%;

text-align: center;

}

input[type=button]{

width:220px;height:auto;font-size:25pt;margin-left: 5px; margin-top:10px;

}

</style>

<script type="text/javascript">

// **map须在DOM加载完毕以后进行构造（比如在window.onload中构造）**

window.onload = function(){

//构造一张地图

map = new Widget.CMap.Map("mapDiv");

//设置地图为矢量图模式

map.setMapType(Widget.CMap.MapType.MAP\_TYTE\_VECTOR);

//设置地图初始的缩放级别为12级

map.setZoom(12);

}  
 /\*\*

\* 发起定位请求

\*/

function requestLocation(){

Widget.CMap.Location.requestMyLocation(map);

}

/\*\*

\* 定位完成后回调，用户自定义事件处理

\*/

Widget.CMap.Location.onMyLocationComplete = function(point){

alert("point-lat:" + point.lat + " lng:" + point.lng);

}

</script>

</head>

<body>

<div id="mapDiv" style="text-align:center;width:400px; height:500px; background-color:#306090;">

</div>

<div style="width:100%;text-align: center">

<input type="button" onclick="requestLocation()"

value="定位"/>

</div>

</body>

</html>

之后补充config.xml并打包，生成的MapDemo.wgt（[点击下载](http://s2.cmribrowser.com/WebAppLib/%20MapDemo.wgt)）即实现显示一张宽度为480像素，高度为620像素，缩放级别为12级、矢量图模式的地图：



**开发者需要注意：**

1. **Map必须在DOM加载完毕以后进行构造。**
2. **网页的meta中不能指定width和initial-scale，否则可能存在地图位置不正确的情况。**

<meta align="center" name="viewport" content="~~width=device-width, initial-scale=1~~" />

1. Widget支付功能开发

无线城市集中平台可统一提供支付能力（支付基地、支付宝等），形成支付服务包，以WAP支付页面方式供各省调用，如果集中平台提供的支付服务满足各省应用需求，则建议各省在应用改造时，直接调用集中平台引入的支付服务，便于运营数据、交易数据的统计分析。 如果集中平台提供的支付服务不满足各省应用需求，各省可使用原有应用支付方式。目前集中平台已引入支付基地支付能力，调用说明见附件《无线城市服务子系统支付服务调用说明》。

鉴于Widget应用离线运行的特点（即Widget应用的代码在客户端而非在Web服务器），直接在Widget应用内部实现支付页面跳转（支付页面在服务器上）的方式可能会存在一些用户体验上的问题。为解决该问题，无线城市客户端中的Widget引擎提供了支付相关的API接口供Widget应用调用，这些接口将支付页面显示和跳转的工作传递给无线城市客户端内置的浏览器引擎来完成，从而保证Widget应用本身的体验。Widget引擎提供的支付功能JavaScript API请参考附录API文档——CPay对象。

**注：通过Widget引擎提供的支付功能API来实现支付页面跳转的方式为可选方式，且只能在无线城市客户端上使用。Widget应用开发人员也可以直接在Widget应用内部实现支付页面跳转。**

**a）使用支付页面跳转方式进行支付示例**

若Widget应用若需要支付能力，首先要在config.xml文件中做出feature声明，如下:

<feature name="cpay/>

在Widget中调用支付页面跳转比较简单，只需要在Widget中调用两个API接口Widget.CPay.urlPay(payActionUrl)和Widget.CPay.onUrlPayFinished()。

调用示例代码如下：

//在需要进行支付的地方调用urlPay()函数，然后页面将调转到具体的支付页面

//**payActionUrl是符合支付能力要求的支付url地址，需要应用自行生成**

Widget.CPay.urlPay(payActionUrl);

//当用户支付完成后onUrlPayFinished()函数将会被自动调用

//因此，支付完成之后的操作，可以在该函数内实现

Widget.CPay.onUrlPayFinished = function() {

//在此处实现支付完成之后需要进行的操作

}

1. Q & A
2. 如何通过JavaScript实现Widget应用退出

由于iOS平台没有“返回”键，这就要求Widget应用需要自己实现“返回”（如下图所示）。在应用首屏，“返回”键一般代表退出本应用。



为实现widget应用退出，可使用Widget.close()函数：

//点击退出键时

Widget.close();

1. 如何解决跨域访问

在网页开发中，为了提高用户体验，很多情况下都会采用异步数据请求，这时候就必然会使用到XmlHttpRequest，简称XHR。以异步请求<http://baidu.com>为例，代码如下：

var Xhr=new XMLHttpRequest();

Xhr.open(‘GET’,’http://baidu.com’,true)；

Xhr.send();

在Chrome或Safari浏览器中运行该代码，会发现该请求会失败。原因是我们本地网页的域是<http://127.0.0.1>, 而请求的网址处于域<http://baidu.com>。由于他们不处于同一个域，因此会请求失败，这就是平时所讲的“跨域访问”。Widget在运行时，网页代码在本地，属于域<http://127.0.0.1>，因此在进行异步数据请求时，也会失败。

如果要解决这种跨域访问的问题，需要对请求的网址所在的服务器进行修改。如果拥有该服务器的控制权限，可以按照以下方法修改，就可以解决跨域问题。

解决方法：修改Web服务器端配置，使得返回页面的Header中Access-Control-Allow-Origin项为“\*”，具体代码如下：

Response.AddHeader(" Access-Control-Allow-Origin", "\*");

原理：浏览器在接收到服务器返回信息时，会检查Header中的Access-Control-Allow-Origin项，它的值表示请求内容所允许的域。现在我们把服务器返回的所有页面的Header中Access-Control-Allow-Origin项设置为“\*”，表明该返回信息允许所有域访问。以上面异步访问baidu.com为例，原本服务器端返回的网页Header中指定只允许域<http://baidu.com>，但是经过我们在Web服务器的修改以后，服务器端返回的网页的Header改变为允许所有域。通过这种修改就解决了跨域访问请求失败的问题。

1. 附录：资源汇总

|  |
| --- |
| * Widget运行客户端   <http://s2.cmribrowser.com/WebAppLib/WidgetClient_V1.2.apk> |
| * [BAE WebApp开发框架](http://s2.cmribrowser.com/WebAppLib/index.html)（请使用Safari或Chrome打开）   <http://s2.cmribrowser.com/WebAppLib/index.html> |
| * HelloWidget.wgt样例程序   <http://s2.cmribrowser.com/WebAppLib/HelloWidget.wgt>   * MessageSample.wgt样例程序   <http://s2.cmribrowser.com/WebAppLib/MessageSample.wgt>   * ScreenAdapt\_1.wgt样例程序   <http://s2.cmribrowser.com/WebAppLib/ScreenAdapt_1.wgt>   * MapDemo.wgt样例程序<http://s2.cmribrowser.com/WebAppLib/MapDemo.wgt> |

1. 附录：Feature声明信息

|  |  |
| --- | --- |
| 能力扩展API对象 | Feature声明信息 |
| AccelerometerInfo | http://jil.org/jil/api/1.1/accelerometerinfo |
| AddressBookItem | http://jil.org/jil/api/1.1/addressbookitem |
| ApplicationTypes | http://jil.org/jil/api/1.1.5/applicationtypes |
| AudioPlayer | http://jil.org/jil/api/1.1/audioplayer |
| Camera | http://jil.org/jil/api/1.1.2/camera |
| Device | http://jil.org/jil/api/1.1/device |
| DeviceStateInfo | http://jil.org/jil/api/1.1/devicestateinfo |
| Message | http://jil.org/jil/api/1.1/message |
| MessageTypes | http://jil.org/jil/api/1.1/messagetypes |
| Messaging | http://jil.org/jil/api/1.1/messaging |
| Multimedia | http://jil.org/jil/api/1.1/multimedia |
| PIM | http://jil.org/jil/api/1.1.1/pim |
| PositionInfo | http://jil.org/jil/api/1.1/positioninfo |
| Widget | http://jil.org/jil/api/1.1/Widget |
| CMap及其下属对象 | cmap |
| CPay | cpay |

1. 附录：Widget能力扩展API文档

## Widget对象

Widget对象是所有能力扩展API的调用入口，其下包括Messaging、Device等其他对象，同时Widget对象也定义了一些方法和事件处理句柄，主要实现跟widget应用相关的一些功能，比如widget应用偏好设置等。

Widget对象的调用属于静态调用，不需构造，如：Widget.openURL("http://www.10086.cn");

**Feature声明：**

1. **常规功能权限声明：http://jil.org/jil/api/1.1/widget**
2. **单点登录权限声明：singlesign，并设置<needwcitySSO>标签**

|  |  |
| --- | --- |
| **方法列表** | |
| <String> | **preferenceForKey**(<String> key)  根据key取对应的preference值。preference值通过setPreferenceForKey来设置 |
| void | **setPreferenceForKey**(<String> preference, <String> key)  setPreferenceForKey用于设置一对key-preference值 |
| void | **openURL**(<String> url)  启动浏览器，打开指定的网页 |
| void | **onWakeup**()  指定widget应用被从后台唤醒时的事件处理句柄 |
| <int> | **getDisplayAreaWidth**()  获取Widget应用视图可视区域宽度,会随引擎视图的大小改变 |
| <int> | **getDisplayAreaHeight**()  获取Widget应用视图可视区域高度,会随引擎视图的大小改变 |
| <double> | **getDisplayScale**()  获取Widget应用视图缩放比例 |
| <String> | **getSSOData**(<String> key)  根据key获取对应的用户信息  注：本函数用于无线城市应用单点登录。 |
| <String> | **getTicket**()  获取T-Ticket身份凭证。Widget应用可以通过T-Ticket身份凭证在应用服务器平台进行身份验证  注：本函数用于无线城市应用单点登录。 |
| void | **close**()  退出Widget应用 |

|  |
| --- |
| **方法详细介绍** |

### preferenceForKey()

<String> **preferenceForKey**(<String> key)

该方法根据key取对应的preference值，preference值通过setPreferenceForKey来设置。如果传入的key不存在，则返回空字符串

**参数：**

key –标识符

**返回：**

key对应的preference值

### setPreferenceForKey()

void **setPreferenceForKey**(<String> preference, <String> key)

该方法用于设置一对key-preference值，该值在设置之后是被持久保存的：只要widget应用不被卸载，这一对值就不会删除。当然，如果widget开发人员认为某一对key – preference值已经不再需要了，可以通过设置preference值为null的办法删除这一对值：Widget.setPreferenceForKey(null, "city");

**参数：**

preference – 待设置的preference值

key - preference标识符

**返回：**

无

### openURL()

void **openURL**(<String> url)

openURL的输入参数是一个url字符串，该方法可以启动Browser，并打开相应的网页，如：

var testURL = "http://www.10086.cn";

Widget.openURL(testURL);

**参数：**

url – 网页url字符串

**返回：**

无

### onWakeup()

void **onWakeup**()

该方法指定widget应用被从后台唤醒时的事件处理句柄。

**参数：**

无

**返回：**

无

### getDisplayAreaWidth()

<int> **getDisplayAreaWidth**()

获取Widget应用视图可视区域宽度，会随手机屏幕的大小改变。

**参数：**

无

**返回：**

Widget应用视图可视区域宽度，单位px

### getDisplayAreaHeight()

<int> **getDisplayAreaHeight**()

获取Widget应用视图可视区域高度，会随手机屏幕的大小改变。

**参数：**

无

**返回：**

Widget应用视图可视区域高度，单位px

### getDisplayScale()

<double> **getDisplayScale**()

获取Widget应用视图缩放比例

**参数：**

无

**返回：**

Widget应用视图缩放比例。如果缩放比例小于0.00001则返回1.0

### getSSOData()

<String> **getSSOData**(<String> key)

根据key获取对应的用户信息。如果key对应的值不存在，则返回空字符串

注：本函数用于无线城市应用单点登录。

**参数：**

key - 用户信息标识符。可被获取的key列表如下：

mobilenumber：表示手机号码

mail：表示邮箱地址

username：用户名

userstatus：用户状态

areacode：用户归属地区域

name：姓名

sex：性别

birthday：生日

address：通讯地址

postcode：邮编

interests：兴趣

profession：职业

married：是否已婚

havechildren：是否有子女

salary：收入区间

school：学历

nationality：国籍

folk：民族

polity：政治面貌

faith：宗教

**返回：**

用户信息值。如果key对应的值不存在，则返回空字符串

### getTicket()

<String> **getTicket**()

获取T-Ticket身份凭证，Widget应用可以通过该T-Ticket身份凭证在应用服务器平台进行身份验证。如果未登录，则返回空字符串。

注：本函数用于无线城市应用单点登录。

**参数：**

无

**返回：**

T-Ticket身份凭证。如果未登录，则返回null

### close()

void **close**()

退出Widget应用。

**参数：**

无

**返回：**

无

## Device对象

Device对象主要实现跟终端相关的一些功能，比如获得手机的应用程序列表、手机震动等。

Device对象位于Widget对象之下，属于静态调用，调用方式为Widget.Device。

**Feature声明：**[**http://jil.org/jil/api/1.1/device**](http://jil.org/jil/api/1.1/device)

|  |  |  |
| --- | --- | --- |
| **方法列表** | | |
| Array<String> | **getAvailableApplications**()  返回系统中可通过widget启动的应用程序列表 | |
| void | **launchApplication**(<String>application,<String>startParameter)  用指定的参数启动一个应用 | |
| void | **vibrate**(<Number> durationSeconds)  使手机震动指定的时间 | |
| <String> | **getDeviceId**()  获取手机设备的唯一标识 | |
| <String> | **getNetworkConnectionType**()  获取当前网络连接类型，可能值为"2G", "3G", "WIFI" | |
| <String> | | **getOSPlatform** ()  获取操作系统平台 |
| <String> | | **getOSVersion** ()  获取操作系统版本号 |

|  |
| --- |
| **方法详细介绍** |

### getAvailableApplications()

Array<String> **getAvailableApplications**()

该方法返回系统中可通过widget启动的应用程序列表，列表中每一个列表元素都是一个应用的名称，应用类型范围为ApplicationTypes中定义的类型

**参数**：

无

**返回：**

系统中可通过Widget启动的应用程序列表

### launchApplication()

void **lanuchApplication**(<String> application, <String> startParameter)

该方法通过给定的启动参数启动一个手机本地的应用。第一个参数是应用的名称，需要从getAvailableApplications的返回结果中去取。

**参数**：

application - 启动的应用程序

startParameter – 启动应用时可带的参数

**返回：**

无

### vibrate()

void **vibrate**(<Number> durationSeconds)

该方法使手机震动指定的时长。

**参数**：

durationSeconds – 指定手机震动的时长，单位为秒

**返回：**

无

### getDeviceId()

<String> **getDeviceId**()

获取手机设备的唯一标识。（Android：IMEI。iOS：PushID）

**参数**：

无

**返回：**

手机设备的唯一标识

### getNetworkConnectionType()

<String> **getNetworkConnectionType**()

获取当前网络连接类型

**参数**：

无

**返回：**

当前网络连接类型，可能值为"2G", "3G", "WIFI"，如果没有网络连接则返回空字符串

### getOSPlatform()

<String> **getOSPlatform**()

获取操作系统平台。

**参数：**

无

**返回：**

操作系统平台。如果是Android，则返回“Android”；是iOS，则返回“iOS”

### getOSVersion()

<String> **getOSVersion**()

获取操作系统版本号。

注：需要获取操作系统平台后参考以下信息：

Android平台下得到系统SDK版本号，其值定义参考Build.VERSION\_CODES：http://developer.android.com/reference/android/os/Build.VERSION\_CODES.html。

iOS平台下得到系统版本号，其值定义类似于“5.1.1”。

**参数：**

无

**返回：**操作系统版本号

## ApplicationTypes对象

ApplicationTypes包括了 Device.launchApplication所需的应用类型值值。

ApplicationTypes对象位于Device对象之下，属于静态调用，调用方式为Widget.Device.ApplicationTypes。

**Feature声明：http://jil.org/jil/api/1.1.5/applicationtypes**

|  |  |
| --- | --- |
| **属性列表** | |
| <String> | **ALARM**  表示设备上的闹钟应用 |
| <String> | **BROWSER**  表示设备上的浏览器应用 |
| <String> | **CALCULATOR**  表示设备上的计算器应用 |
| <String> | **CALENDAR**  表示设备上的日历应用 |
| <String> | **CAMERA**  表示设备上的照相机应用 |
| <String> | **CONTACTS**  表示设备上的联系人应用 |
| <String> | **FILES**  表示设备上的文件管理应用 |
| <String> | **GAMES**  表示设备上的游戏管理或者游戏文件夹 |
| <String> | **Mail**  表示设备上的邮箱应用 |
| <String> | **MEDIAPLAYER**  表示设备上的多媒体播放应用 |
| <String> | **MESSAGING**  表示设备上的信息应用 |
| <String> | **PHONECALL**  表示设备上的Dialer拨号盘应用 |
| <String> | **PICTURES**  表示设备上的照片预览功能 |
| <String> | **PROG\_MANAGER**  表示设备上的程序管理应用 |
| <String> | **SETTINGS**  表示设备上的设置应用 |
| <String> | **TASKS**  表示设备上的‘To do list’应用 |

|  |
| --- |
| **属性详细介绍** |

### ALARM

<String> **ALARM**

表示设备上的闹钟应用。

### BROWSER

<String> **BROWSER**

表示设备上的浏览器应用。

### CALCULATOR

<String> **CALCULATOR**

表示设备上的计算器应用。

### CALENDAR

<String> **CALENDAR**

表示设备上的日历应用。

### CAMERA

<String> **CAMERA**

表示设备上的照相机应用。

### CONTACTS

<String> **CONTACTS**

表示设备上的联系人应用。

### FILES

<String> **FILES**

表示设备上的文件管理应用。

### GAMES

<String> **GAMES**

表示设备上的游戏管理或者游戏文件夹应用。

### MAIL

<String> **MAIL**

表示设备上的邮箱应用。

### MEDIAPLAYER

<String> **MEDIAPLAYER**

表示设备上的多媒体播放应用。

### MESSAGING

<String> **MESSAGING**

表示设备上的信息应用。

### PHONECALL

<String> **PHONECALL**

表示设备上的dialer拨号应用。

### PICTURES

<String> **PICTURES**

表示设备上的照片浏览应用。

### PROG\_MANAGER

<String> **PROG\_MANAGER**

表示设备上的程序管理应用。

### SETTINGS

<String> **SETTINGS**

表示设备上的设置应用。

### TASKS

<String> **TASKS**

表示设备上的‘To-do list’待办事项应用。

## Positioninfo对象

Positioninfo主要提供手机位置相关的属性。

Positioninfo对象位于Device对象之下，属于静态调用，调用方式为Widget.Device.Positioninfo。

**Feature声明：http://jil.org/jil/api/1.1/positioninfo**

|  |  |
| --- | --- |
| **属性列表** | |
| <Number> | **accuracy**  水平位置精度，单位 米，数据类型为浮点数数值 |
| <Number> | **altitude**  海拔，单位 米，数据类型为浮点数数值，使用WGS84数据标准 |
| <Number> | **altitudeAccuracy**  垂直位置精度，单位 米，数据类型为浮点数数值 |
| <Number> | **cellID**  用于定位的移动Cell ID |
| <Number> | **Latitude**  纬度，单位 度，数据类型为浮点数数值，使用WGS84数据标准 |
| <Number> | **Longitude**  经度，单位 度，数据类型为浮点数数值，使用WGS84数据标准 |
| <Date> | **timestamp**  确定位置时的时间 |

|  |
| --- |
| **属性详细介绍** |

### accuracy

<Number> **accuracy**

水平位置精度，单位 米，数据类型为浮点数数值

### altitude

<Number> **altitude**

海拔，单位 米，数据类型为浮点数数值，使用WGS84数据标准

### altitudeAccuracy

<Number> **altitudeAccuracy**

垂直位置精度，单位 米，数据类型为浮点数数值

### cellID

<Number> **cellID**

用于定位的移动Cell ID

### latitude

<Number> **latitude**

纬度，单位 度，数据类型为浮点数数值，使用WGS84数据标准

### longitude

<Number> **longitude**

经度，单位 度，数据类型为浮点数数值，使用WGS84数据标准

### timeStamp

<Date> **timestamp**

确定位置时的时间

## DeviceStateInfo对象

DeivceStateInfo主要包含获取手机位置信息等方法，

DeivceStateInfo对象在Device对象之下，属于静态调用，调用方式为Widget.Device.DeviceStateInfo。

**Feature声明：http://jil.org/jil/api/1.1/devicestateinfo**

|  |  |
| --- | --- |
| **方法列表** | |
| void | **requestPositionInfo**(<String> method)  查询手机位置，该方法需要和onPositionRetrieved事件处理句柄配合使用 |
| void | **onPositionRetrieved**(<PositionInfo> locationinfo, <String> method)  这是一个事件处理句柄，由requestLocationInfo()触发 |

|  |
| --- |
| **方法详细介绍** |

### requestPositionInfo()

**void requestPositionInfo** (<String>method)

该方法用来查询手机位置，该方法需要和onPositionRetrieved事件处理句柄配合使用：requestPositionInfo是异步方法，本身并不返回位置查询结果，而是在得到查询结果后触发onPositionRetrieved事件处理，而且查询结果（一个PositionInfo对象）也会传入onPositionRetrieved事件处理句柄中。

**参数**：

method – 获得位置的方法，可取的值包括：cellid，gps、agps

**返回：**

无

### onPositionRetrieved()

void **onPositionRetrieved(<PositionInfo> locationinfo, <String> method)**

该方法是一个事件处理句柄，需要和requestPositionInfo配合使用。

**参数**：

locationinfo – 位置查询结果

method – 获得位置的方法

**返回：**

无

## AccelerometerInfo对象

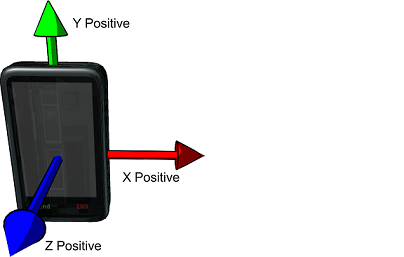
AccelerometerInfo定义了手机设备在x、y、z轴上的加速度，共三个属性。

AccelerometerInfo对象位于DeviceStateInfo对象下层，属于静态调用，调用方式为Widget.Device.DeviceStateInfo.AccelerometerInfo。

**Feature声明：http://jil.org/jil/api/1.1/accelerometerinfo**

|  |  |
| --- | --- |
| **属性列表** | |
| <Number> | **xAxis**  xAxis是手机设备在x轴上的加速度 |
| <Number> | **yAxis**  yAxis是手机设备在y轴上的加速度 |
| <Number> | **zAxis**  zAxis是手机设备在z轴上的加速度 |

下图表示了手机设备x、y、z轴的位置：

****

注意AccelerometerInfo对象的三个属性的单位都是m/s^2，数据类型为浮点数数值。三个属性都是只读，没有set操作。

|  |
| --- |
| **属性详细介绍** |

### xAxis

<Number> **xAxis**

xAxis是手机设备在x轴上的加速度，由手机设备在x轴上的重力值计算出来x轴的方向是手机设备向右正方向

### yAxis

<Number> **yAxis**

yAxis是手机设备在y轴上的加速度，由手机设备在y轴上的重力值计算出来，y轴的方向是手机设备向上正方向

### zAxis

<Number> **zAxis**

zAxis是手机设备在z轴上的加速度，由手机设备在z轴上的重力值计算出来，z轴的方向是手机设备前侧正方向

## Messaging对象

Messaging对象主要实现跟短消息相关的一些功能，比如消息生成、发送等。

Messaging对象位于Widget对象之下，属于静态调用，调用方式为Widget.Messaging。

**Feature声明：http://jil.org/jil/api/1.1/messaging**

|  |  |
| --- | --- |
| **方法列表** | |
| <Message> | **createMessage**(<String> messageType)  创建一条消息 |
| void | **onMessageSendingFailure**(<Message> msg, <string> error)  如果信息发送失败的回调方法 |
| void | **sendMessage**(<Message> msg)  发送一条指定消息 |

|  |
| --- |
| **方法详细介绍** |

### createMessage()

<Message> **createMessage**(<String> messageType)

该方法用于创建一个消息。

**参数：**

messageType – 消息的类型，取值范围参考MessageTypes对象

**返回：**

一个Message对象

### onMessageSendingFailure()

void **onMessageSendingFailure**(<Message> msg, <string> error)

用户自定义信息发送失败时的回调函数。

**参数：**

msg – 发送失败的信息

error –错误信息

**返回：**

无

### sendMessage()

void **sendMessage**(<Message> msg)

该方法用于发送一条指定信息。

**参数：**

msg – 待发送的信息

**返回：**

无

## MessageTypes对象

MessageTypes用于指定消息的类型。

MessageTypes对象在Messaging对象之下，属于静态调用，调用方式为Widget.Messaging.MessageTypes。

**Feature声明：http://jil.org/jil/api/1.1/messagetypes**

|  |  |
| --- | --- |
| **属性列表** | |
| <String> | **SMSMessage**  表示SMS类型 |

|  |
| --- |
| **属性详细介绍** |

### SMSMessage

<String> **SMSMessage**

表示SMS类型

## Message对象

Message对象对应的是一条信息，它包含了消息类型、消息地址、消息体等属性和方法。

Message对象在Messaging对象之下，只能通过Messaging.createMessage方法生成，不可以单独构造。

**Feature声明：http://jil.org/jil/api/1.1/message**

|  |  |
| --- | --- |
| **属性列表** | |
| <String> | body  消息的正文内容 |
| <String> | callbackNumber  消息的回复号码 |
| Array<String> | destinationAddress  消息的目的地址 |
| <boolean> | isRead  消息是否已读 |
| <String> | messageId  消息的ID，是保存在手机上的消息的唯一标识 |
| <boolean> | messagePriority  消息是否高优先级 |
| <String> | messageType  消息类型 |
| <String> | sourceAddress  消息的源地址 |
| <String> | Subject  消息的主题 |
| <Date> | Time  消息的时间 |
| <Number> | validityPeriodHours  消息的有效期 |

|  |  |
| --- | --- |
| **方法列表** | |
| void | **addAddress**(<String> type, <String> address)  addAddress方法用于给消息增加地址 |
| void | **deleteAddress**(<String> type, <String> address)  deleteAddress方法用于删除地址 |

|  |
| --- |
| **属性详细介绍** |

### body

<String> **body**

消息的正文内容

### callbacknumber

<String> **callbackNumber**

消息的回复号码

### destinationAddress

Array<String> **destinationAddress**

消息的目的地址

### isRead

<boolean> **isRead**

消息是否已读

### messageId

<String> **messageId**

消息的编号

### messagePriority

<boolean> **messagePriority**

消息是否高优先级

### messageType

<String> **messageType**

消息的类型，取值范围参考MessageTypes对象

### sourceAddress

<String> **sourceAddress**

消息的源地址

### subject

<String> **subject**

消息的主题

### time

<Date> **time**

消息的时间

### validityPeriodHours

<Number> **validityPeriodHours**

消息的有效期

|  |
| --- |
| **方法详细介绍** |

### addAddress()

void **addAddress** (<String> type, <String> address)

该方法用于给消息添加地址。

**参数：**

type – 添加的地址类型，可能的值包括："destination"

address –被添加的地址

**返回：**

无

### deleteAddress()

void **deleteAddress**(<String> type, <String> address)

该方法用于给消息删除地址。

**参数：**

type – 删除的地址类型，可能的值包括："destination"

address –被删除的地址

**返回：**

无

## Multimedia对象

Multimedia对象主要实现跟多媒体相关的一些功能，比如摄像头、音频播放等。

Multimedia对象位于Widget对象之下，属于静态调用，调用方式为Widget.Multimedia。

**Feature声明：http://jil.org/jil/api/1.1/multimedia**

|  |  |
| --- | --- |
| **属性列表** | |
| <boolean> | isAudioPlaying  查询手机是否正在播放音频 |

|  |  |
| --- | --- |
| **方法列表** | |
| <Number> | **getVolume**()  得到的音频音量值，从0 - 10，其中0表示关闭 |
| void | **stopAll**()  停止当前正在播放的所有音频 |

|  |
| --- |
| **属性详细介绍** |

### isAudioPlaying

<boolean> **isAudioPlaying**

查询手机是否正在播放音频

|  |
| --- |
| **方法详细介绍** |

### getvolume()

<Number> **getVolume**()

该方法用于得到当前音频音量

**参数：**

无

**返回：**

音频音量，为0到10之间的整数值，其中0表示关闭

### stopAll()

void **stopAll**()

该方法用于停止当前正在播放的所有音频。

**参数：**

无

**返回：**

无

## AudioPlayer对象

AudioPlayer定义了一些跟音频播放功能相关的方法。

AudioPlayer对象位于Multimedia对象之下，属于静态调用，调用方式为Widget.Multimedia.AudioPlayer。

**Feature声明：http://jil.org/jil/api/1.1/audioplayer**

|  |  |
| --- | --- |
| **方法列表** | |
| void | **onStateChange**(<String> state)  onStateChange是一个回调函数，在AudioPlayer的播放状态发生变化时被触发 |
| void | **open**(<String> fileUrl)  用于打开一个音频文件 |
| void | **pause**()  让playing状态下的AudioPlayer暂停 |
| void | **play**(<Number> playQty)  play方法可以让opened、stopped、completed状态下的AudioPlayer开始播放 |
| void | **resume**()  resume方法可以让paused状态下的AudioPlayer恢复播放 |
| void | **stop**()  stop方法可以让playing或paused状态下的AudioPlayer停止播放 |

|  |
| --- |
| **方法详细介绍** |

### onStateChange()

void **onStateChange**(<String> state)

onStateChange是一个回调函数，在AudioPlayer的播放状态发生变化时被触发，触发时，当前播放状态将当成参数传入。

下图了AudioPlayer的状态转换情况。



**参数：**

state - 表示当前播放的状态，状态包括："opened", "playing", "stopped", "paused", "completed"

**返回：**

无

### open()

void **open**(<String> fileUrl)

open方法用于打开一个音频文件，做好播放前的准备。

**参数：**

fileUrl - 音频文件的绝对路径或是相对路径。对于绝对路径，open方法支持如下几种格式："file://"、"http://"、"https://"和"rtsp://"；如果文件路径不是以上面的四种前缀开头，那么说明输入参数是相对路径，那么将打开widget安装路径下的某个音频文件

**返回：**

无

### pause()

void **pause**()

pause方法可以让playing状态下的AudioPlayer暂停。

**参数：**

无

**返回：**

无

### play()

void **play**(<Number> playQty)

play方法可以让opened、stopped、completed状态下的AudioPlayer开始播放

**参数：**

playQty – 重复播放次数

**返回：**

无

### resume()

void **resume**()

resume方法可以让paused状态下的AudioPlayer恢复播放

**参数：**

无

**返回：**

无

### Stop()

void **stop**()

stop方法可以让playing或paused状态下的AudioPlayer停止播放

**参数：**

无

**返回：**

无

## Camera对象

Camera对象定义了一些跟手机拍照、摄像等功能相关的方法。

Camera对象位于Multimedia对象之下，属于静态调用，调用方式为Widget.Multimedia.Camera。

**Feature声明：http://jil.org/jil/api/1.1.2/camera**

|  |  |
| --- | --- |
| **方法列表** | |
| void | **captureImage**(<String> fileName, <boolean> lowRes)  使用手机上的摄像头拍摄照片 |
| void | **onCameraCaptured**(<String> fileName)  onCameraCaptured是回调函数，在拍照并成功保存到本地时触发 |
| void | **setWindow**(<Object> domObj)  将视频播放器与一个DOM对象相连接，令此DOM对象作为呈现摄像头在拍照或摄像之前的预览窗 |

|  |
| --- |
| **方法详细介绍** |

### captureImage()

void **captureImage**(<String> fileName, <boolean> lowRes)

captureImage方法用于使用手机上的摄像头拍摄照片

**参数：**

filename - 指定照片的保存路径

lowRes -是否使用低像素模式：如果该参数为true，那么将拍摄低分辨率照片，如果该参数定义为false，将拍摄高分辨率照片

**返回：**

无

### onCameraCaptured()

void **onCameraCaptured**(<String> fileName)

onCameraCaptured是回调函数，在拍照并成功保存到本地时触发。触发时照片保存路径将作为参数传入。

**参数：**

filename - 照片的保存路径

**返回：**

无

### setWindow()

void **setWindow**(<Object> domObj)

setWindow方法可以将视频播放器与一个DOM对象相连接，令此DOM对象作为呈现摄像头在拍照或摄像之前的预览窗。

**参数：**

domObj - 照相机指定预览区域的一个HTML DOM元素

**返回：**

无

## PIM对象

PIM对象主要实现跟个人信息管理相关的功能，比如通信录等。

PIM对象位于Widget对象之下，属于静态调用，调用方式为Widget.PIM。

**Feature声明：http://jil.org/jil/api/1.1/pim**

|  |  |
| --- | --- |
| **方法列表** | |
| <AddressBookItem> | **createAddressBookItem**()  创建联系人 |
| void | **findAddressBookItems**(<AddressBookItem>  comparisonContact, <Number> startInx, <Number> endInx)  搜索联系人 |
| void | **onAddressBookItemsFound**(Array<AddressBookItem> addressBookItemsFound)  联系人搜索的回调函数 |
| <AddressBookitem> | **getAddressBookItem**(<String> id)  根据指定id获取对应的联系人 |
| <Number> | **getAddressBookItemsCount**()  取得联系人的个数 |

|  |
| --- |
| **方法详细介绍** |

### createAddressBookItem()

<AddressBookItem> **createAddressBookItem**()

该方法用于创建一个内容为空的联系人对象

**参数：**

无

**返回：**

一个内容为空的联系人对象

### findAddressBookItems()

void **findAddressBookItems**(<AddressBookItem> comparisonContact, <Number> startInx, <Number> endInx)

该方法用于搜索联系人

**参数：**

comparisonContact – 用于对比搜索的AddressBookItem对象

startInx – 一个整数值，指定索引返回的搜索结果开始。最小值：0

endInx - 一个整数值，指定索引返回的搜索结果结束。最小值：0。**注：(endInx-startInx)最大值为20，对于大于20的输入将以20进行计算**

**返回：**

无

### onAddressBookItemsFound()

void **onAddressBookItemsFound**(Array<AddressBookItem> addressBookItemsFound)

onAddressBookItemsFound是回调函数，搜索联系人成功时触发。触发时搜索结果将作为参数传入。

**参数：**

addressBookItemsFound – 搜索结果数组

**返回：**

无

### getAddressBookItem()

<AddressBookItem> **getAddressBookItem**(<String> id)

该方法用于根据制定id获取联系人

**参数：**

id – 联系人的唯一标识符

**返回：**

返回联系人对象

### getAddressBookItemsCount()

<Number> **getAddressBookItemsCount**()

该方法用于取得联系人的个数

**参数：**

无

**返回：**

联系人个数

## AddressBookItem对象

AddressBookItem对象对应的是一个联系人，它包含了手机号、地址、邮箱等属性和方法。

AddressBookItem对象在PIM对象之下，只能通过PIM.createAddressBookItem方法生成，不可以单独构造。

**Feature声明：http://jil.org/jil/api/1.1/addressbookitem**

|  |  |
| --- | --- |
| **属性列表** | |
| <String> | address  联系人的地址 |
| <String> | addressBookItemId  联系人ID，是联系人的唯一标识 |
| <String> | company  联系人的公司 |
| <String> | eMail  联系人的e-Mail地址 |
| <String> | fullName  联系人的姓名 |
| <String> | homePhone  联系人的家庭电话号码 |
| <String> | mobilePhone  联系人的手机号码 |
| <String> | title  联系人的公司职位 |
| <String> | workPhone  联系人的工作电话号码 |

|  |  |
| --- | --- |
| **方法列表** | |
| <String> | **getAttributeValue**(<String> attribute)  用于取得联系人某个属性的值 |
| Array<String> | **getAvailableAttributes**()  用于取得联系人中有效的属性列表 |
| void | **setAttributeValue**(<String> attribute, <String> value)  用于设置联系人的属性 |
| void | **update**()  用于将一个修改后的联系人信息更新到手机中 |

|  |
| --- |
| **属性详细介绍** |

### address

<String> **address**

address是联系人的地址，数据类型为字符串。

### addressBookItemId

<String> **addressBookItemId**

addressBookItemId是联系人的ID，数据类型为字符串，是保存在手机上的联系人的唯一标识。

### company

<String> **company**

company是联系人的公司，数据类型为字符串。

### eMail

<String> **eMail**

eMail是联系人的e-Mail地址，数据类型为字符串。

### fullName

<String> **fullName**

fullName是联系人的姓名，数据类型为字符串。

### homePhone

<String> **homePhone**

homePhone是联系人的家庭电话号码，数据类型为字符串。

### mobilePhone

<String> **mobilePhone**

mobilePhone是联系人的手机号码，数据类型为字符串。

### title

<String> **title**

title是联系人的公司职位，数据类型为字符串。

### workPhone

<String> **workPhone**

workPhone是联系人的工作电话号码，数据类型为字符串。

|  |
| --- |
| **方法详细介绍** |

### getAttributeValue()

<String> **getAttributeValue**(<String> attribute)

getAttributeValue用于取得联系人某个属性的值。

**参数：**

attribute - AddressBookItem中的一个属性名

**返回：**

联系人该属性的对应值

### getAvailableAttributes()

 Array<String> getAvailableAttributes()

该方法用于取得联系人中有效的属性列表，返回值为一个字符串数组。

**参数：**

无

**返回：**

有效的属性列表

### setAttributeValue()

void **setAttributeValue**(<String> attribute, <String> value)

该方法用于设置联系人的属性值。

**参数：**

attribute - AddressBookItem中的某一个属性名

value - 被分配的一个值

**返回：**

无

### update()

void **update**()

update方法用于将一个修改后的联系人信息更新到手机中。

**参数：**

无

**返回：**

无

## CMap对象

CMap对象定义了一套地图应用开发调用接口，其下包括Map、Point等其他对象，通过这套接口，您可以在自己的widget应用中定制地图功能。

CMap是整个地图API的命名空间，位于Widget对象之下，属于静态调用，调用方式为Widget.CMap。

**Feature声明：cmap**

## Map对象

Map对象是地图基础类，地图实例化入口。如果需要在页面中显示地图则需要先创建Map对象，并将对象关联的页面的div元素。

Map对象位于CMap对象之下，调用方式为Widget.CMap.Map，该对象使用时需要通过构造函数构造。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **属性列表** | |
| <int> | **zoom**  缩放级别，默认值为12 |
| <int> | **type**  地图类型。取值范围参考MapType对象  默认类型为矢量图 |
| <Point> | **center**  地图显示区域的中心点 |

|  |  |
| --- | --- |
| **方法列表** | |
| <Map> | **Map**(<String> divId)  构造器，用于创建一个地图视图 |
| void | **setCenter**(<Point > center)  设置地图的中心点，用于切换当前显示地图位置 |
| void | **setZoom**(<Number > zoom)  设置地图的缩放级别。默认缩放级别为12  矢量地图的取值范围[3,20]，栅格地图的取值范围为[3,18] |
| void | **setMapType**(<MapType> mapType)  设置地图类型 |
| void | **showTraffic**(<boolean> show)  设置是否显示交通路况 |
| void | **setZoomControlEnabled**(<boolean> enabled)  设置地图上是否显示内置的缩放按钮。（注：仅支持Android平台） |
| void | **addOverlay**(<Overlay> overlay)  将覆盖物添加到地图中，一个覆盖物实例只能向地图中添加一次 |
| void | **removeOverlay**(<Overlay> overlay)  从地图中移除覆盖物，如果覆盖物从未被添加到地图中，则该移除不起任何作用 |
| void | **clearOverlays**()  清除地图上所有覆盖物 |
| void | **startTapPoint**()  触发获取地图坐标事件 |
| void | **onTapPointGot**()  用户点击地图获取坐标操作完成后，系统自动回调函数 |

|  |
| --- |
| **属性详细介绍** |

### zoom

<int> zoom

缩放级别，默认值为12

### type

<int> type

地图类型。取值范围参考MapType对象

默认类型为普通视图

### center

<Point> center

地图显示区域的中心点

|  |
| --- |
| **方法详细介绍** |

### Map()

<Map> **Map**(<String> divId)

该方法为构造方法，用于创建Map对象。使用方式为：var map = new Widget.CMap.Map("mapDivId");

**参数**：

divId - 为html页面中对应显示地图的div区域的id

**返回：**

Map对象

### setCenter()

void **setCenter** (<Point > center)

该方法设置地图的中心点，用于切换当前显示地图中心位置

**参数**：

center - Point类型的中心点坐标

**返回：**

无

### setZoom()

void **setZoom** (<int> zoom)

设置地图的缩放级别。默认缩放级别为12

注：矢量地图的取值范围[3,20]，栅格地图的取值范围为[3,18]。

**参数**：

zoom - 地图的缩放级别

**返回：**

无

### setMapType()

void **setMapType**(<MapType> mapType)

设置地图类型

**参数**：

mapType - 地图类型，详见MapType

**返回：**

无

### showTraffic()

void **showTraffic**(<boolean> enabled)

设置是否显示交通路况

**参数**：

enabled –true表示显示交通路况；false表示不显示交通路况

**返回：**

无

### setZoomControlEnabled()

void **setZoomControlEnabled**(<boolean> enabled)

设置地图上是否显示内置的缩放按钮。（注：仅支持Android平台）

**参数**：

enabled – true表示显示内置缩放按钮；false表示不显示内置缩放按钮

**返回：**

无

### addOverlay()

void **addOverlay**(<Overlay> overlay)

将覆盖物添加到地图中，一个覆盖物实例只能向地图中添加一次

**参数**：

overlay – 待增加的覆盖物对象，可以包括Marker对象、Circle对象、Polyline对象、Polygon对象

**返回：**

无

### removeOverlay()

void **removeOverlay**(<Overlay> overlay)

从地图中移除覆盖物，如果覆盖物从未被添加到地图中，则该移除不起任何作用

**参数**：

overlay – 待移除的覆盖物对象

**返回：**

无

### clearOverlays()

void **clearOverlays**()

清除地图上所有覆盖物

**参数**：

无

**返回：**

无

### startTapPoint()

void **startTapPoint**()

触发获取地图坐标事件。当用户获取坐标以后系统自动回调map对象的getTapPoint(point)函数。

**参数**：

无

**返回：**

无

### onTapPointGot()

void **onTapPointGot**(point)

用户点击地图获取坐标操作完成后，系统自动回调函数。

**参数**：

point - Point对象。用户点击地图获取的坐标。

**返回：**

无

## MapType对象

MapType对象是地图类型的属性表示，位于CMap对象之下，该对象调用属于静态调用，方式为Widget.CMap.MapType。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **属性列表** | |
| <Number> | **MAP\_TYPE\_SATELLITE**  常量，卫星视图 |
| <Number> | **MAP\_TYPE\_BITMAP**  常量，栅格视图 |
| <Number> | **MAP\_TYPE\_VECTOR**  常量，矢量视图 |

|  |
| --- |
| **属性详细介绍** |

### MAP\_TYPE\_SATELLITE

<Number> MAP\_TYPE\_SATELLITE

常量，卫星视图

### MAP\_TYPE\_BITMAP

<Number> MAP\_TYPE\_BITMAP

常量，栅格视图

### MAP\_TYPE\_VECTOR

<Number> MAP\_TYPE\_VECTOR

常量，矢量视图

## Point对象

Point对象是用于表示地图元素的坐标，通常在对地图上元素进行定位时使用。

Point对象位于CMap对象之下，调用方式为Widget.CMap.Point，该对象使用时需要通过构造函数构造。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **属性列表** | |
| <Number> | **lat**  位置坐标的纬度 |
| <Number> | **lng**  位置坐标的经度 |

|  |  |
| --- | --- |
| **方法列表** | |
| <Point> | **Point**(<Number>lat, <Number>lng)  构造器，用于创建一个坐标点 |
| <boolean> | **equals** (<Point> pt)  判断两个坐标点是否相等，当且仅当两点的经度和纬度均相等时才认为坐标相等 |

|  |
| --- |
| **属性详细介绍** |

### lat

<Number> lat

位置坐标的纬度，双精度数值。有效范围为[-180.0,180.0]。

### lng

<Number> lng

位置坐标的经度，双精度数值。有效范围为[-80.0,80.0]。

|  |
| --- |
| **方法详细介绍** |

### Point()

<Point> **Point**(<Number>lat, <Number>lng)

该方法为构造方法，用于创建Point对象

**参数**：

lat - 纬度

lng - 经度

**返回：**

Point对象

### equals()

<boolean> **equals** (<Point > pt)

判断两个坐标点是否相等，当且仅当两点的经度和纬度均相等时才认为坐标相等。

**参数**：

pt – 待比较的坐标点

**返回：**

true - 如果两坐标经纬度相同

false - 如果两坐标经纬度不相同

## Overlay对象

Overlay对象为地图覆盖物基类，无具体属性和方法，不可以构造对象使用。其子类有Marker、Circle、Polyline、Polygon。

**Feature声明：cmap**

## Marker对象

Marker对象是用于在地图上显示的标点，继承至Overlay。

Marker对象位于CMap对象之下，调用方式为Widget.CMap.Marker，该对象使用时需要通过构造函数构造。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **方法列表** | |
| <Marker> | **Marker**(<Point>point)  构造器，用于创建一个Marker标点 |
| void | **setLabel**(<String> label)  设置标点上显示的文字内容 |
| void | **setBubbleMsg**(<String> bubbleMsg)  设置标点关联的气泡提示信息 |
| void | **setIcon**(<String> iconPath)  设置标点的图标的图片资源路径 |
| void | **onBubbleClick**()  定义标点关联的气泡的点击事件 |
| void | **closeBubble**()  关闭标点关联的气泡提示信息 |

|  |
| --- |
| **方法详细介绍** |

### Marker()

<Marker> **Marker**(<Point>point)

该方法为构造方法，用于创建Marker对象

**参数**：

point – 标点的坐标

**返回：**

Marker对象

### setLabel()

void **setLabel** (<String> label)

该方法用于设置标点上显示的文字内容。

**参数**：

label - 文字内容

**返回：**

无

### setBubbleMsg()

void **setBubbleMsg** (<String> bubbleMsg)

该方法用于设置标点关联的气泡提示信息

**参数**：

bubbleMsg - 标点关联的气泡提示信息

**返回：**

无

### setIcon()

void **setIcon**(<String> iconPath)

该方法用于设置标点的图标的图片资源路径。iconPath是以Widget应用的根目录为起点的相对路径描述，比如Widget应用目录结构如下:

│ config.xml

│ icon.png

│ index.html

├─css

│ global.css

└─img

marker.png

要设置Marker的图标资源为marker.png，则在js中调用代码marker.setIcon("img/marker.png")即可。

**参数**：

iconPath - 图标的图片资源路径

**返回：**

无

### onBubbleClick()

void **onBubbleClick**()

定义标点关联的气泡的点击事件

**参数**：

无

**返回：**

无

### closeBubble()

void **closeBubble**()

关闭标点关联的气泡提示信息

**参数**：

无

**返回：**

无

## Circle对象

Circle对象是用于在地图上显示圆圈，继承至Overlay。

Circle对象位于CMap对象之下，调用方式为Widget.CMap.Circle，该对象使用时需要通过构造函数构造。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **方法列表** | |
| <Circle> | **Circle**(<Point> center,<Number> radius)  构造器，用于创建一个Circle对象 |
| void | **setStrokeStyle**(<Number> strokeStyle)  设置圆圈的边框颜色 |
| void | **setStrokeOpacity**(<Number> strokeOpacity)  设置圆圈的边框颜色透明度 |
| void | **setFillStyle**(<Number> fillStyle)  设置圆圈的填充颜色 |
| void | **setFillOpacity**(<Number> fillOpacity)  设置圆圈的填充颜色透明度 |
| void | **setLineWidth**(<Number> lineWidth)  设置圆圈边框的宽度 |

|  |
| --- |
| **方法详细介绍** |

### Circle()

<Circle> **Circle**(<Point> center,<Number> radius)

该方法为构造方法，用于创建Circle对象

**参数**：

center – 中心点的坐标

radius – 半径，单位为米

**返回：**

Circle对象

### setStrokeStyle()

void **setStrokeStyle**(<Number> strokeStyle)

设置圆圈的边框颜色

**参数**：

strokeStyle - 圆圈的边框颜色，如0x0000ff

**返回：**

无

### setStrokeOpacity()

void **setStrokeOpacity**(<Number> strokeOpacity)

设置圆圈的边框颜色透明度

**参数**：

strokeOpacity - 圆圈的边框颜色透明度，取值0-100

**返回：**

无

### setFillStyle()

void **setFillStyle**(<Number > fillStyle)

设置圆圈的填充颜色

**参数**：

fillStyle - 圆圈的填充颜色，如0x0000ff。

**返回：**

无

### setFillOpacity()

void **setFillOpacity** (<Number > fillOpacity)

设置圆圈的填充颜色透明度

**参数**：

fillOpacity - 圆圈的填充颜色透明度，取值0-100

**返回：**

无

### setLineWidth()

void **setLineWidth** (<Number > lineWidth)

该方法用于设置圆圈边框的宽度

**参数**：

lineWidth - 圆圈边框的宽度

**返回：**

无

## Polyline对象

Polyline对象是用于在地图上显示折线，继承至Overlay。

Polyline对象位于CMap对象之下，调用方式为Widget.CMap.Polyline，该对象使用时需要通过构造函数构造。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **方法列表** | |
| <Polyline> | **Polyline**(<Array> points)  用于创建Polyline对象 |
| void | **setStrokeStyle**(<Number > strokeStyle)  设置线条颜色 |
| void | **setStrokeOpacity**(<Number > strokeOpacity)  设置线条颜色透明度 |
| void | **setLineWidth**(<Number > lineWidth)  设置线条宽度 |
| void | **setPath**(<Array> points)  设置折线路径 |

|  |
| --- |
| **方法详细介绍** |

### Polyline()

<Polyline> **Polyline**(<Array> points)

该方法为构造方法，用于创建Polyline对象

**参数**：

points – 标点的坐标数组

**返回：**

Polyline对象

### setStrokeStyle()

void **setStrokeStyle**(<Number > strokeStyle)

设置线条颜色

**参数**：

strokeStyle - 线条颜色，如0x0000ff。

**返回：**

无

### setStrokeOpacity()

void **setStrokeOpacity**(<Number > strokeOpacity)

设置线条颜色透明度

**参数**：

strokeOpacity - 线条颜色透明度，取值0-100。

**返回：**

无

### setLineWidth()

void **setLineWidth**(<Number > lineWidth)

设置线条宽度

**参数**：

lineWidth -设置线条宽度。

**返回：**

无

### setPath()

void **setPath**(<Array> points)

设置折线路径

**参数**：

points -折线的顶点坐标数组。

**返回：**

无

## Polygon对象

Polygon对象是用于在地图上显示多边形，继承至Overlay。

Polygon对象位于CMap对象之下，调用方式为Widget.CMap.Polygon，该对象使用时需要通过构造函数构造。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **方法列表** | |
| <Polygon> | **Polygon**(<Array> points)  用于创建Polygon对象 |
| void | **setStrokeStyle**(<Number> strokeStyle)  设置边框颜色 |
| void | **setStrokeOpacity**(<Number> strokeOpacity)  设置边框颜色透明度 |
| void | **setFillStyle**(<Number> fillStyle)  设置圆圈的填充颜色 |
| void | **setFillOpacity**(<Number> fillOpacity)  设置填充颜色透明度 |
| void | **setLineWidth**(<Number> lineWidth)  设置边框的宽度 |
| void | **setPath**(<Array> points)  设置多边形路径 |

|  |
| --- |
| **方法详细介绍** |

### Polygon()

Polygon **Polygon**(<Array> points)

该方法为构造方法，用于创建Polygon对象。

**参数**：

points – 标点的坐标数组

**返回：**

Polygon对象

### setStrokeStyle()

void **setStrokeStyle**(<Number> strokeStyle)

设置边框颜色。

**参数**：

strokeStyle -边框颜色，如0x0000ff。

**返回：**

无

### setStrokeOpacity()

void **setStrokeOpacity**(<Number> strokeOpacity)

设置边框颜色透明度。

**参数**：

strokeOpacity -边框颜色透明度，取值0-100。

**返回：**

无

### setFillStyle()

void **setFillStyle**(<Number> fillStyle)

设置填充颜色。

**参数**：

fillStyle -填充颜色，如0x0000ff。

**返回：**

无

### setFillOpacity()

void **setFillOpacity**(<Number> fillOpacity)

设置填充颜色透明度。

**参数**：

fillOpacity -填充颜色透明度，取值0-100。

**返回：**

无

### setLineWidth()

void **setLineWidth** (<Number> lineWidth)

该方法用于设置边框的宽度。

**参数**：

lineWidth - 边框的宽度

**返回：**

无

### setPath()

void **setPath** (<Array> points)

设置多边形路径。

**参数**：

points -多边形的顶点坐标数组。

**返回：**

无

## PoiSearch对象

PoiSearch对象用于地图上的兴趣点检索功能。

PoiSearch对象通过CMap对象静态调用，调用方式为Widget.CMap. PoiSearch。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **方法列表** | |
| void | **poiQuery**(<Map> map, <String> keyWord, <Number>city, <Number>size)  周边检索，根据中心点、半径与检索词进行检索 |
| void | **onPoiSearchComplete**(<PoiPagedResult> poiPagedResult)  用户自定义POI搜索完成的回调函数，结果传入一个PoiPagedResult对象中 |

|  |
| --- |
| **方法详细介绍** |

### poiQuery()

void **poiQuery**(<Map> map,

<String> keyWord,

<Number> city,

<Number> size)

周边检索，根据地图显示区域、检索词、城市代码进行检索。如果参数map已定义，则查询地图显示区域内的POI，否则如果城市代码已定义，则查询为该城市（地区）内的POI，如上述两个条件均未定义，则范围为全国。

**参数**：

map –Map对象，用于指定搜索区域（即在Map显示区域范围内搜索）

keyWord –查询字符串

city – 待查询城市（地区）的电话区号，如：北京：010，上海：021

size – 查询结果中每页的数量。**注：每页数量最大值为20，对于大于20的输入将以20进行计算**

**返回：**

无

### onPoiSearchComplete()

void **onPoiSearchComplete**(<PoiPagedResult> poiPagedResult)

用户自定义POI搜索完成的回调函数，搜索结果传入一个PoiPagedResult对象中。

**参数**：

poiPagedResult – POI搜索结果对象。

**返回：**

无

## PoiPagedResult对象

PoiPagedResult对象用于放置POI搜索结果，即PoiSearch完成搜索后通过回调onPoiSearchComplete方法传递给用户的结果。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **方法列表** | |
| <Number> | **getPageCount**()  获取搜索结果总页数 |
| Array<Position> | **getPage**(<Number> pageNumber)  返回页码pageNumber对应的结果[Position对象数组]，页码pageNumber从1开始计数  如果该页已查询过，则返回结果，否则联网访问 |

|  |
| --- |
| **方法详细介绍** |

### getPageCount()

<Number> **getPageCount**()

获取搜索结果总页数。

**参数**：

无

**返回：**

Number - 结果总页数

### getPage()

 Array<Position> **getPage**(<Number> pageNumber)

返回页码pageNumber对应的结果[Position对象数组]，页码pageNumber从1开始计数。

如果该页已查询过，则返回结果，否则联网访问。

**参数**：

pageNumber – 页码

**返回：**

Array<Position> - 页码pageNumber对应的结果

## Position对象

Position对象是POI搜索的单个结果点。

Position对象位于CMap对象之下，调用方式为Widget.CMap.Position。该对象只能在POI搜索时由Widget引擎内部生成，开发者无需自行构造。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **方法列表** | |
| <String> | **getName()**  获取POI的名称 |
| <Point> | **getPoint()**  获取POI的经纬度坐标 |
| <String> | **getAddress()**  获取POI的地址 |
| <String> | **getPostcode()**  获取POI的邮编信息[行政区划代码] |
| <String> | **getPhone()**  获取POI的电话号码 |

|  |
| --- |
| **方法详细介绍** |

### getName()

<String> **getName()**

获取POI的名称

**参数**：

无

**返回：**

POI的名称

### getPoint()

<Point> getPoint()

获取POI的经纬度坐标

**参数**：

无

**返回：**

POI的经纬度坐标

### getAddress()

<String>getAddress()

获取POI的地址

**参数**：

无

**返回：**

POI的地址

### getPostcode()

<String> getPostcode()

获取POI的邮编信息[行政区划代码]

**参数**：

无

**返回：**

POI的邮编信息[行政区划代码]

### getPhone()

<String> getPhone()

获取POI的电话号码

**参数**：

无

**返回：**

POI的电话号码

## RouteSearch对象

RouteSearch对象用于地图上的路线规划搜索功能。

RouteSearch对象通过CMap对象静态调用，调用方式为Widget.CMap. RouteSearch。

注：如果存在多次路线规划搜索，则最后一次搜索结果会把前次搜索结果覆盖，因此对于每次的搜索结果需要开发者自行处理。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **属性列表** | |
| <Number> | **TransitMode**  线路模式:公交。用于查询时设定线路模式 |
| <Number> | **DrivingMode**  线路模式:自驾/步行。用于查询时设定线路模式 |

|  |  |
| --- | --- |
| **方法列表** | |
| void | **routeQuery**(<Map> map, <Number> mode, <String> ptStart, <String> ptEnd)  路线搜索 |
| void | **onRouteSearchComplete**(<RouteResult> routeResult)  用户自定义ROUTE搜索完成的回调函数，搜索结果传入RouteResult对象中 |

|  |
| --- |
| **属性详细介绍** |

### TransitMode

<Number> TransitMode

线路模式:公交。用于查询时设定线路模式。

### DrivingMode

<Number> DrivingMode

线路模式: 自驾/步行。用于查询时设定线路模式。

|  |
| --- |
| **方法详细介绍** |

### routeQuery()

void **routeQuery**(<Map> map,<Number> mode,<String> ptStart ,<String> ptEnd)

路线搜索。

**参数**：

map –Map对象

mode – 线路模式

ptStart – 起点关键字

ptEnd – 终点关键字

**返回：**

无

### onRouteSearchComplete()

void **onRouteSearchComplete**(<RouteResult> routeResult)

用户自定义ROUTE搜索完成的回调函数，搜索结果传入RouteResult对象中。

**参数**：

routeResult – 线路搜索结果。

**返回：**

无

## RouteResult对象

RouteResult对象用于放置ROUTE搜索结果，即RouteSearch完成搜索后通过回调onRouteSearchComplete方法传递给用户的结果。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **方法列表** | |
| <Number> | **getRouteResultSize**()  获取路线搜索结果总数 |
| <Route> | **getRoute**(<Number> routeNumber)  返回routeNumber对应的结果[一个Route对象]，routeNumber从0开始计数 |

|  |
| --- |
| **方法详细介绍** |

### getRouteResultSize()

<Number> **getRouteResultSize** ()

获取路线搜索结果总数。

**参数**：

无

**返回：**

Number – 搜索到的路线数量

### getRoute()

<Route> **getRoute**(<Number> routeNumber)

返回routeNumber对应的结果[一个Route对象]，routeNumber从0开始计数。

**参数**：

routeNumber – 路线索引

**返回：**

无

## Route对象

Route对象是ROUTE搜索的单个结果点。

Route对象位于CMap对象之下，调用方式为Widget.CMap.Route。该对象只能在路线规划搜索时由Widget引擎内部生成，开发者无需自行构造。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **方法列表** | |
| <Point> | **getStartPoint()**  路线起点经纬度坐标 |
| <Point> | **getEndPoint()**  路线终点经纬度坐标 |
| Array<Segment> | **getSegments()**  路线的路段数组 |
| <Number> | **getDistance()**  路线距离，单位为米 |
| void | **show**(<Map> map)  调用系统绘制线路，在map上显示路线 |
| void | **removeFromMap**(<Map>map)  取消map上显示的线路 |

|  |
| --- |
| **方法详细介绍** |

### getStartPoint()

<Point> getStartPoint()

获取路线起点经纬度坐标

**参数**：

 无

**返回：**

 路线起点经纬度坐标

### getEndPoint()

<Point> getEndPoint()

获取路线终点经纬度坐标

**参数**：

 无

**返回：**

 路线终点经纬度坐标

### getSegments()

Array<Segment> getSegments()

获取路线的路段数组

**参数**：

 无

**返回：**

 路线的路段数组

### getDistance()

<Number> getDistance()

获取路线距离，单位为米

**参数**：

 无

**返回：**

 路线距离，单位为米

### show()

void **show**(<Map> map)

调用系统绘制线路，在map上显示路线

**参数**：

map - Map对象

**返回：**

 无

### removeFromMap()

 void **removeFromMap**(<Map> map)

取消map上显示的线路

**参数**：

map - Map对象

**返回：**

 无

## Segment对象

Segment对象是用于定义路线规划搜索后返回的路线对象中若干路段的对象，其中含有步行和公交路段。

Segment对象位于CMap对象之下，调用方式为Widget.CMap.Segment。该对象只能在路线规划搜索时由Widget引擎内部生成，开发者无需自行构造。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **方法列表** | |
| <String> | **getConsumeTime()**  此路段的消耗时间 |
| <Point> | **getFirstPoint()**  此路段的首个形状点坐标 |
| <Point> | **getLastPoint()**  此路段的最后一个形状点坐标 |
| <Number> | **getLength()**  长度，单位为米 |
| Array<Point> | **getShapes()**  此路段的坐标点集合 |
| <String> | **getFirstStationName()**  公交线路始发站的名称。(若为自驾/步行路线则该字段未定义) |
| <String> | **getLastStationName()**  公交线路终点站的名称。(若为自驾/步行路线则该字段未定义) |
| <String> | **getLineName()**  公交线路的名称。(若为自驾/步行路线则该字段未定义) |
| <String> | **getOffStationName()**  下车站的站名。(若为自驾/步行路线则该字段未定义) |
| Array<String> | **getPassStops()**  上车站的站名数组。(若为自驾/步行路线则该字段未定义) |
| <Number> | **getStopNumber()**  返回从上车站到下车站的车站数目，这个数目包含上车站与下车站。(若为自驾/步行路线则该字段未定义) |
| <String> | **getActionDescription()**  返回此路段的行动描述，如左转、右转等。(若为公交路线则该字段未定义) |
| <String> | **getRoadName()**  返回此路段对应的道路名称。(若为公交路线则该字段未定义) |

|  |
| --- |
| **方法详细介绍** |

### getConsumeTime()

<String> getConsumeTime()

此路段的消耗时间。

**参数**：

 无

**返回：**

 路段的消耗时间

### getFirstPoint()

<Point> getFirstPoint()

此路段的首个形状点坐标。

**参数**：

 无

**返回：**

 此路段的首个形状点坐标

### getLastPoint()

<Point> getLastPoint()

此路段的最后一个形状点坐标。

**参数**：

 无

**返回：**

此路段的最后一个形状点坐标

### getLength()

<Number> getLength()

长度，单位为米。

**参数**：

 无

**返回：**

 长度，单位为米

### getShapes()

Array<Point> getShapes()

此路段的坐标点集合。

**参数**：

 无

**返回：**

 此路段的坐标点集合

### getFirstStationName()

<String> getFirstStationName()

公交线路始发站的名称。(若为自驾/步行路线则该字段未定义)

**参数**：

 无

**返回：**

 公交线路始发站的名称

### getLastStationName()

<String> getLastStationName

公交线路终点站的名称。(若为自驾/步行路线则该字段未定义)

**参数**：

 无

**返回：**

 公交线路终点站的名称

### getLineName()

<String> getLineName()

公交线路的名称。(若为自驾/步行路线则该字段未定义)

**参数**：

 无

**返回：**

公交线路的名称

### getOffStationName()

<String> getOffStationName()

下车站的站名。(若为自驾/步行路线则该字段未定义)

**参数**：

 无

**返回：**

 下车站的站名

### getPassStops()

Array<String> getPassStops()

途经车站的站名数组。(若为自驾/步行路线则该字段未定义)

**参数**：

 无

**返回：**

 途经车站的站名数组

### getStopNumber()

<Number> getStopNumber()

从上车站到下车站的车站数目，这个数目包含上车站与下车站。(若为自驾/步行路线则该字段未定义)

**参数**：

 无

**返回：**

 从上车站到下车站的车站数目

### getActionDescription()

<String> getActionDescription()

返回此路段的行动描述，如左转、右转等。(若为公交路线则该字段未定义)

**参数**：

 无

**返回：**

 此路段的行动描述，如左转、右转等

### getRoadName()

<String> getRoadName()

返回此路段对应的道路名称。(若为公交路线则该字段未定义)

**参数**：

 无

**返回：**

 此路段对应的道路名称

## Location对象

Location对象用于定位功能。

Location对象通过CMap对象静态调用，调用方式为Widget.CMap. Location。

**Feature声明：cmap**

|  |  |
| --- | --- |
| **方法列表** | |
| void | **requestMyLocation**(<Map>map)  请求定位 |
| void | **onMyLocationComplete**(<Point> myLocation)  用户自定义定位完成的回调函数，定位结果传入一个Point对象中 |

|  |
| --- |
| **方法详细介绍** |

### requestLocation()

void **requestLocation**(<Map>map)

请求定位

**参数**：

map – 地图对象

**返回：**

无

### onMyLocationComplete()

void **onMyLocationComplete** (<Point> myLocation)

用户自定义定位完成的回调函数，定位结果传入一个Point对象中。用户在JS中重写方法Widget.CMap.Location.onMyLocationComplete() {}，实现自定义定位完成后的事件处理。

**参数**：

myLocation – 定位成功后的坐标

**返回：**

无

## CPay对象

CPay对象定义了用于支付功能开发的调用接口，通过这套接口，您可以在自己的widget应用中定制支付功能。

CPay是整个支付API的命名空间，位于Widget对象之下，该对象调用属于静态调用，调用方式为Widget.CPay。

**Feature声明：cpay**

|  |  |
| --- | --- |
| **方法列表** | |
| void | **urlPay**(<String> payUrl)  传入带支付参数的支付地址进行支付 |
| void | **onUrlPayFinished**()  支付完成后用户自定义事件处理 |

|  |
| --- |
| **方法详细介绍** |

### urlPay()

void **urlPay**(<String> payUrl)

传入带支付参数的支付地址进行支付。

**参数**：

payUrl - 带支付参数的支付地址

**返回：**

无

### onUrlPayFinished()

void **onUrlPayFinished**()

支付完成后用户自定义事件处理。用户在JS中重写方法Widget.CPay.onUrlPayFinished() {}，实现自定义支付完成后的事件处理。

**参数**：

无

**返回：**

无