移动混合开发框架①

徐隆龙, 李 莹, 白 静

(国网电力科学研究院 北京中电普华信息技术有限公司, 北京 100192)

摘 要: 针对现有混合开发框架在开发便利性和应用性能方面的不足, 研究了以 HTML5 开发为基础的移动混合开发框架的问题. 结合 HTML5 跨平台特性、打包机制、多窗口管理机制、原生插件开发技术等, 实现了一套自有的混合开发框架. 本混合开发框架为开发带来了便利, 提升了开发效率, 同时, 混合引擎中独特实现的 Widget 应用管理以及窗口管理模式也改善了混合应用的性能, 提升了用户体验. 我公司实现的企业级移动解决方案表明该框架性能较好, 易于使用, 具有较好的商业价值.

关键词: 移动开发; 原生应用; 混合应用; Web 应用; HTML5

Hybrid Architecture for Mobile Development

XU Long-Long, LI Ying, BAI Jing

(Beijing China Power Information Technology Co. Ltd, State Grid Electric Power Research Institute, Beijing 100192, China)

Abstract: In order to satisfy the deficiency of the existing hybrid development framework in terms of convenience and performance, this paper proposed a new hybrid development framework. With the cross-platform features of HTML5, packing mechanism, multi-window management mechanism, and native plugin development technology, this paper implemented the hybrid development framework. The hybrid development framework brought convenience and enhanced the development efficiency. What's more, the unique Widget application management mechanim and window management mechanim in the engine improved the performance of hybrid applications as well as users' experience. The enterprise mobile solutions indicated that the framework had good commercial value, which was ease to use and had good performance.

Key words: mobile development; native application; hybrid application; Web application; HTML5

随着移动互联网的发展, BYOD ("Bring Your Own Device", 自带设备上班) 成为企业移动应用的关键词. 它旨在提高企业信息传递的及时性、准确性、安全性, 完善企业核心数据的虚拟化、异地化管理, 实现企业员工的移动化、智能化办公需求. 在移动信息化风潮扑面而来时, 移动环境特点和技术特点在企业信息化建设过程中, 成为了各企业所面临的新型问题. 而目前国家电网 99%的系统为传统的企业内部应用, 需要运行在固定场所、固定网络环境和固定设备上, 在对外服务和对内办公效率的提升上, 形成了一定阻碍. 为了适应新的形势, 做为国家电网信息化建设的承包单位之一, 普华集团各业务中心及子公司也正在积极应对企业移动应用方面的需求, 提出开展移动应用开发框架的研究.

操作系统是连接应用程序和设备的桥梁, 操作系统在智能手机和移动 APP 的发展中扮演了核心角色. 由于各移动操作系统的不兼容性, 原生模式开发的应用(Native App)只能运行在特定的设备上, android 和 iOS 上的应用需要单独开发, 这给应用开发商带来了较大的困扰. 针对这种困境, 基于浏览器的 HTML5 标准迅速崛起, 基于 HTML5 的混合开发模式(Hybrid App)也得到了快速的发展[1]. 原生开发模式和混合开发模式都有成功的案例, 但是面对企业众多业务系统和多终端多平台的难题, 众多企业, 尤其是传统行业,

① 收稿时间:2014-03-26;收到修改稿时间:2014-05-08

倾向采用混合开发模式.

基于HTML5开发的Hybrid App特性更接近Native App, 但是因为同时使用了网页开发语言, 所以开发成本和难度比 Native App 小很多, 在开发成本上接近于Web App. 所以说, Hybrid App 兼具 Native App 和 Web App 两者的诸多优点. 本文的目标旨在基于混合移动应用技术, 建设一套跨平台的移动应用开发框架.

1 研究现状

开发一个移动应用程序曾经是一个巨大的工程, 它需要大量的准备和开发时间, 新技术和流程的出现把这个时间大大的降低[2]. Hybrid App 的开发通常借助一些框架来实现[3], PhoneGap、Titanium、JQuery Mobile、Sencha Touch 等备受推崇, 但这几种框架各有利弊. PhoneGap 和 Titanium 框架是借助 Native 应用的特性实现的不同移动 OS 平台的原生框架, 相对 Titanium 框架来说, PhoneGap 框架更为广大开发人员所熟知, 并且一些知名的移动开发工具中也对 PhoneGap 框架做了集成, 如 IBM 的 WorkLight 开发工具和 SAP 的 SMP 平台中都集成了 PhoneGap. JQuery Mobile、Sencha Touch 框架是运行在移动端的 Web 框架, 随着 WebKit 浏览体验的升级, JQuery Mobile、Sencha Touch 等也获得了较好的发展. 同 PhoneGap 一样, 在 IBM 和 SAP 的产品中也同样对 JQuery Mobile、Sencha Touch 做了集成.

1.1 PhoneGap 和 Titanium 框架

PhoneGap(现已命名为 apache Cordova)是基于 Web 开发技术创建跨平台移动应用的开源开发框架. PhoneGap 旨在解决两个问题[4]: 在不同 OS 的移动设备上像本地应用一样运行同一个 Web 应用; 实现 JavaScript 和本地 API 之间的调用和通信. 为实现这两大目标, PhoneGap 提供的功能大致如下:

(1) 提供统一的打包平台, 用户可通过 dreamweaver 或者在线打包平台等进行统一打包, 经打包之后 Web 应用就被嵌在了原生应用的 “盒子”中, 进而运行在终端设备上.

(2) 将本地功能如摄像头、传感器、文件调用等封装成为 JavaScript 调用, 开发者就像使用标准 JavaScript 类库一样. 另外, PhoneGap 还允许开发者按照一定的规则使用平台指定的本地语言编写功能.

PhoneGap 为跨平台开发打开了一扇新的大门, 为移动应用的开发提供了一种更高效的开发模式. 但是它并没有提供 IDE 开发环境, 开发需要借助第三方移动应用开发框架来开发应用.

Titanium 框架同 PhoneGap 框架原理相似, 都是通过传统 Web 开发并用 JS 调用本地设备能力. PhoneGap 并不通过 JS 暴露本地的 UI 接口, 而 Titanium 则有完整的本地 UI 接口, 通过 JS 调用就能够获得本地一样的 UI, 通过这些 UI JS 接口 Titanium 的应用能够比 PhoneGap 的应用看起来更加接近本地. 另外, Titanium 有以下缺点, 其某些 API 是平台相关的, 使用这些 API 会降低这些应用的跨平台能力; Titanium 不兼容标准 JavaScript, 开发者需要按照它规定的语法书写应用代码, 这给开发造成了一定的难度[5]. 基于以上两点, Titanium 框架并没有取得像 PhoneGap 框架一样的成功.

1.2 JQuery Mobile 和 Sencha Touch 框架

JQuery Mobile 和 Sencha Touch 都是基于 HTML5 的移动开发 JS 框架, 提供了一系列的页面组件, 支持事件处理等, 使用这两个框架构建的应用程序具有良好的用户体验. 二者虽然可实现同样的功能, 但 JQuery Mobile 源于 JQuery, 采用传统 div 布局机制, 是基于传统的 Web 开发模式, 入门简单, Sencha Touch 源于 ExtJS, 完全采用 javascript 布局机制, 是基于组件的开发模式, 需要开发者有较好的 javascript 基础.

JQuery Mobile 和 Sencha Touch 都只是前端框架, 开发者使用其开发的应用是纯 Web 应用, 并不具有本地能力.

1.3 存在问题

从 1.1 和 1.2 章节的分析可知, 目前业界现有框架只关注了原生功能调用(如 PhoneGap 和 Titanium 框架)或者只关注了页面展现方式的实现(如 JQuery Mobile 和 Sencha Touch), 没有一个完整的可供开发者直接使用的开发框架. 由于原生框架和前端框架在功能上是互补的, 似乎将 PhoneGap 和 JQuery Mobile 等框架结合使用能帮助开发者完美解决混合开发的问题, 但事实是开发者要学习两种框架的使用方式, 要自己做集成, 这增加了开发难度. 在国内, 能掌握两种开发框架并熟练使用的开发者并不多.

将两种框架相结合的开发方式, 除学习成本较高之外, 开发出来的混合应用也往往不能让人满意, 这种方式实现的多 HTML 页面间的切换导致应用内存释

放较慢, 内存消耗量飙升, 应用性能较差.

针对这种开发现状, 我司决定打造一套特有的适合开发者快速使用的混合开发框架. 基于原生开发技术, 本文提出了一套完整的移动混合开发框架, 综合了 PhoneGap 和 JQuery Mobile 框架的优点, 并且有较好的用户体验.

2 混合开发框架架构

2.1 技术路线

混合开发的研究大体有两个方向. 由于 Android 系统运行的应用基于 java 开发, 而 iOS 系统运行的应用基于 Objective_C 开发, 如果想真正的实现跨平台, 最直观的做法是发明第三种语言, 这种语言可以解释成 java 语言, 同时可以解释成 Objective_C 语言. 目前来看, 这种方式的研究进展缓慢, 还没有形成统一的行业标准.

第二种方式是基于 HTML5 的跨平台特性来实现的, 这也是当前各种跨平台框架的一致技术原理. 众所周知, 在 Android (iOS)设备中内置了基于 WebKit 内核的浏览器, 因此移动设备可以加载 Web 应用[6], 如图 1 所示, Android 手机通过 Android 浏览器来访问 Web 应用, iPhone 手机中的 safari 浏览器同样用于访问 Web 应用.

图 1 移动浏览器内核

除了提供基于 WebKit 内核的浏览器之外, 在 Android 和 iOS 的 SDK 中还封装了 WebView(UIWebView)视图组件, 通过该视图类暴露 WebKit 的接口给应用的开发者. 如图 1 中, 在 Android 和 iOS 的 Native 应用中, 开发者使用该视图对象加载 Web 网页, 原生应用就可以显示和处理请求网络资源. 这种本质上是原生应用, 但却使用 WebView 和 UIWebView 来展示网页的应用, 我们称之为混合应用, 又由于展示的这些网页资源是跨 Android 和 iOS 平台的, 所以我们又称之为跨平台应用. 所以, 基于这种方式的跨平台开发技术解决的关键问题是提供网页资源的运行容器, 我们称之为引擎, 即实现对 WebView(UIWebView)的管理. 关于引擎的具体作用及实现方式见 2.3 节.

如果只加载和处理 Web 资源, 那么这种应用并非真正意义的混合应用, 混合应用与 Web 应用的区别是支持 Web 功能和原生功能的相互嵌套. 从实现机制上讲, Web 功能和原生功能的相互嵌套即 JS 代码和原生代码的相互调用. 除用于展示网页外, WebView 和 UIWebView 还提供一种扩展机制, 该扩展允许 JavaScript 代码和本地代码的相互调用(跟所有浏览器内核一致, WebKit 内核也分为 WebCore 模块和 JSCore 模块, 从 JSCore 模块的实现可知, JavaScript 对象是通过 C 和 C++来实现的, 而 iOS 的本地语言 Objective_C 扩充至 C, 并且 Objective_C 的编译器完全兼容 C, 因此 Objective_C 和 JavaScript 的相互调用是可行的. 同样的 Java 可以通过 JNI 方式与 C 交互, 这使得 Java 和 JavaScript 的相互调用成为可能). 为了达到功能的混合, 方便 JavaScript 和本地代码的相互调用, 利用 WebView 和 UIWebView 提供的扩展机制, 将本地功能按照一定规范, 做成插件的形式, 具体的实现方式见 2.3.3 节.

最后, 利用打包技术将 HTML5 代码分别嵌入到不同的 Native 应用中供混合代码运行在不同的平台.

综上所述, 本文所设计和实现的混合应用的系统架构图如图 2 所示. 其中 Android 和 iOS 基础应用包含引擎和插件, 它们通过调用 WebKit 内核在 Api 层提供的视图组件 WebView 和 UIWebView 来管理网页代码.

图 2 混合应用系统架构图

2.2 整体框架设计

由章节 2.1 可知, 通过 WebView 和 UIWebView 可以将 HTML5 网页嵌入到 android 和 iOS 原生应用中.

跨平台开发整体结构如图 3 所示. 跨平台开发以打包平台为核心, 通过打包平台生成 Android 或 iOS 应用. 打包平台其实是将网页包和原生插件、引擎和签名文件(原生应用的特性, 在 Android 中为使用 KeyTool 生成的 KeyStore 文件, 在 iOS 中为苹果开发者账号申请的发布证书或者企业证书)做组合之后, 然后调用原生编译环境和应用导出功能, 分别生成 Android 和 iOS 的安装包.

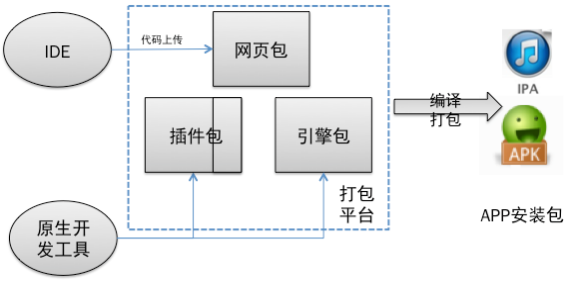

图 3　混合开发整体设计

引擎包为混合开发框架的核心部分, 其主要作用是解析网页包和插件包, 提供网页代码和原生代码的运行环境.

从本质上将, 引擎包为可独立运行的原生应用, 所以引擎包分为 android 的引擎包和 iOS 的引擎包, 并分别由官方提供的开发工具来完成, 具体实现见 2.3 节.

2.3 混合开发引擎设计

引擎的主要作用是提供 HTML5 应用的运行环境, 并且提供一套完整的页面窗口管理机制. 另外, 为弥补 Web 开发的不足, 引擎还要负责提供原生功能的调用机制.

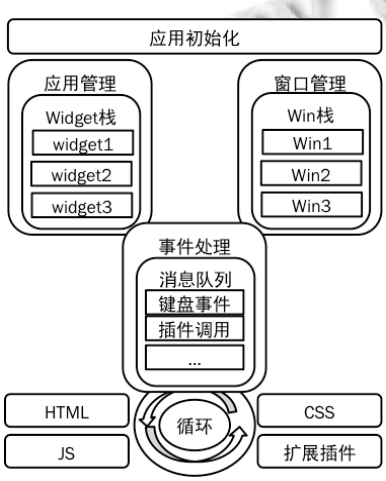

图 4　混合引擎系统架构图

应用运行时, 引擎的处理过程如图 4 所示. 引擎首先通过应用管理器去加载 Web 应用, 通过窗口管理器控制窗口的生命周期、维护各窗口之间的关系. 同时, 引擎中还会维护一个消息队列, 用于事件处理, 主要包括用户的操作事件和插件的调用事件等. 而插件调用事件的处理通过引擎中插件管理模块来实现的. 引擎的功能架构图如图 5 所示.

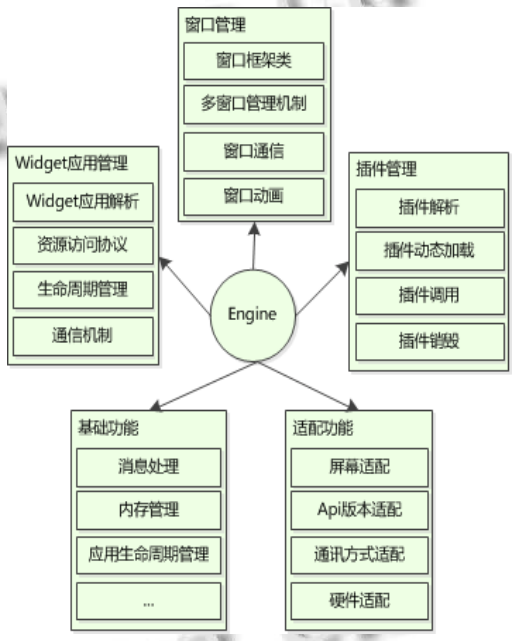

图 5　混合引擎功能架构图

从图 5 可以看出, 引擎共提供 5 大功能, 对各功能的具体介绍在 2.3.1-2.3.5 节实现. 由于引擎是独立的 iOS 应用和 Android 应用, 下述 2.3.1-2.3.5 节中实现的功能都是 Android 和 iOS 单独实现, 但是, 在功能上和实现机制上 Android 引擎和 iOS 引擎是相似的, 简单期间, 本文只以 Android 系统为例做详细介绍.

2.3.1 Widget 应用管理

为了实现 Web 代码的共享, 在本文设计的混合引擎中提出了 Widget 应用的概念. Widget 应用是指具有完整功能的 HTML5 应用, 它是一组 HTML 页面以及 JS 和 CSS 文件的集合. 一个混合应用中至少有一个 Widget 应用, 我们称之为主 Widget, 可以有 0 个或者多个子 widget, 引擎对 widget 的管理机制如图 6 所示.

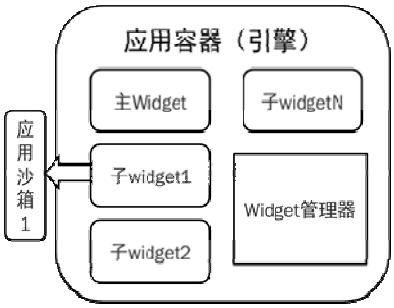

图 6 应用管理机制

(1)Widget 应用解析: Widget 解析是指引擎解析并加载应用, 为实现这一功能, 为每个 Widget 应用额外增加一个配置文件. 该配置文件主要记录 widget 的唯一标识 ID、Widget 的首页面 URL、Widget 更新地址、开发者信息等. 引擎通过配置文件找到 widget 应用对应的首页面进行加载.

(2)资源访问协议: 每个 Widget 应用都具有唯一的沙箱, 引擎通过对固定协议地址的转化可以保证每个 widget 应用可访问到本 widget 沙箱中的资源, 不可访问其他 widget 沙箱中的资源.

(3)Widget 生命周期管理: 引擎通过维持一个容器栈来对 Widget 应用进行管理. 混合应用中主 widget 的加载是在应用启动后引擎初始化时加载的. 引擎根据主 widget 的配置文件建立与主 widget ID 对应的页面容器(与 Android 中的布局相对应). 子 Widget 是通过主 widget 打开和关闭的, 分别对应容器的创建、进出栈以及子容器的添加和移除操作.

(4)通信机制: 此处的通信机制是指主 widget 和子 widget 之间参数的传递, 当主 widget 打开子 widget 时可向子 widget 传递参数, 子 widget 处理参数; 当子 widget 关闭时, 可回传参数到主 widget.

2.3.2 窗口管理机制

(1)窗口框架管理: 窗口框架采用主浮窗口机制, 一个窗口包含 top、main 和 bottom 三部分, 其中 top 和 bottom 部分为可选部分, 但位置固定, PopOver 为覆盖在主窗口的 HTML 网页, 可在任意位置. 通过 Android 帧布局、线性布局和相对布局的嵌套以及 WebView 的组合, 实现如图 7 所示的窗口框架.

(2)多窗口管理机制: 引擎支持多窗口管理机制, 对窗口的管理模仿 Android 对 Activity 的管理机制来实现, 具体实现方式为:

在引擎中维持一个窗口视图栈, 应用的主窗口位于栈底;

当从窗口 A 跳转到窗口 B 时, 如果窗口 B 已经被打开, 则将窗口 B 移动到栈顶部, 如果窗口 B 未被打开, 则新建 B 窗口视图, 并置于栈顶;

当关闭当前窗口时, 则将窗口从栈中弹出, 并且将该窗口放置于垃圾窗口中.

(3)窗口通信: 各主窗口以及主窗口和任一浮动窗口之间可通过 js 相互通信, 即一个窗口可执行另一个窗口的 JS 方法或调用另一个窗口定义的 JS 属性. 从实现上讲, 各窗口之间的通信其实就是一个 Webview 可执行另一个 Webview 中定义的 JS 代码, 这一执行机制借助 WebKit 中 JSCore 模块解析和执行 JS 方法来实现.

(4)窗口动画: 由于主窗口、浮动窗口和兄弟窗口都是由 Webview 组件实现的, 并且 Webview 组件都都位于线性布局或帧布局之中, 所以各窗口的打开和关闭都可通过给布局添加动画效果来实现, 主要包括旋转、平移、缩放、透明度改变等动画效果.

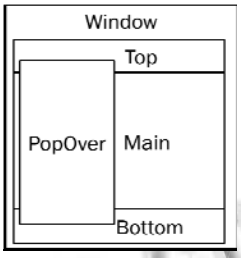

图 7 窗口框架图

2.3.3 插件管理

混合开发中, 通常将设备能力调用或其他必须用 Native 来实现的功能包装成 js 对象供 Web 调用, 即插件.

同对 Widget 应用的管理机制一样, 为便于引擎解析插件, 插件除包含原生代码之外还包含配置文件, 用来建立 java 对象和 JS 对象的映射关系. 插件配置文件如下所示:

表 1 插件配置文件示例

```
plugin.xml
<plugins>
<!-- plugin Camera -->
<plugin
    className="org. plugin.CameraPlugin"
    pluginName="uexCamera" >
```

```
    <method name="openCamera" />
    <method name="… " />
</plugin>
</plugins>
```

引擎对插件的管理主要包括如下几个步骤:

(1)插件解析: 引擎读取并解析插件配置文件获取所有需要注册的插件信息, 通过反射机制得到要注册的插件类.

(2)插件加载: 由插件类得到插件实例, 利用 Webview 提供的插件注册机制, 建立 java 对象和 js 对象的映射关系, 此处的调用关系如图 8 所示:

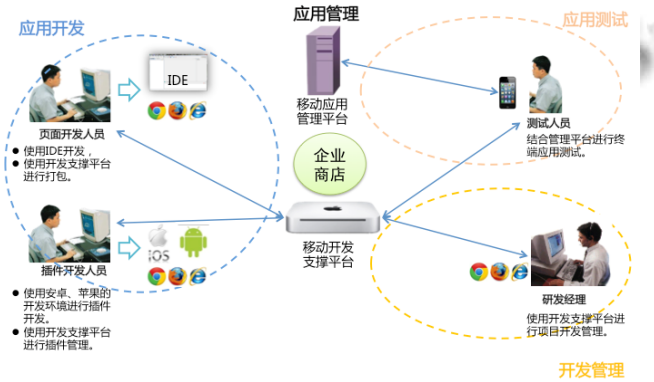

图 8　普华移动开发平台部署结构图

(3)插件调用: 为便于引擎对插件方法的统一管理和统一的参数机制, 建立插件方法和上述 js 对象的映射关系, 即开发人员通过调用 js 方法就可以直接执行原生代码. 并且, 当原生方法执行完成之后, 通过回调机制可以将执行结果同步或异步的回传给调用者.

(4)插件销毁: 由于插件的注册时通过 Webview 来完成的, 当 Webview 被关闭时, 插件会被自动销毁, 释放相应资源, 销毁插件对应的对象.

另外, 为满足用户的个性化需求, 增加框架的扩展性, 混合开发框架中除提供常用基础插件之外, 引擎还支持开发者自定义扩展插件.

2.3.4 适配机制

引擎提供的适配机制主要包括屏幕适配、Api 版本适配、通讯方式适配、硬件适配等.

屏幕适配机制是通过 CSS 和引擎的配合完成的. 引擎获取设备的屏幕密度, 通过 WebView 提供的接口为网页上字体和图片的展示提供最适合的单位, 用户在编写网页代码时统一使用该单位.

引擎通过获取设备支持的 Api 版本, 从而判断所调用的 Api, 并对 WebView 等需要行个性化设置的功能进行统一管理.

对于联网方式的适配涉及到混合应用的管理机制问题, 此处不再介绍.

硬件适配主要针对一些特殊的功能进行适配, 比如蓝牙调用和传感器调用等跟硬件相关的功能.

2.3.5 基础功能

基础功能主要包括应用的内存管理、消息处理、缓存管理、应用生命周期管理等. 此功能为所有 Android 和 iOS 应用中所必备的功能, 其实现机制跟技术原理跟跨平台开发和混合开发藕合度较小, 在此不再赘述.

3　应用案例

为了快速实现企业移动信息化建设需要, 我公司实现了一个以企业商店为中心, 以混合开发为基础的集应用开发、发布、管理等功能为一体的企业级移动解决方案. 各系统之间的协作关系为:

(1)项目研发经理通过开发支撑平台新建应用, 并为该应用添加开发者;

(2)Native 开发人员开发应用特有插件, 上传到开发支撑平台;

(3)Web 前端开发人员编写页面代码, 对于纯 web 无法实现的功能, 调用插件 api, 编写完成之后利用 svn 等代码管理工具上传代码到开发支撑平台.

(4)测试人员通过开发支撑平台打包应用, 安装测试.

(5)应用测试通过之后由运维人员将应用上传到企业商店中.

(6)应用上线之后, 运维人员通过应用管理平台对应用做消息推送、升级管理、运营监控等.

本文提出的混合开发框架主要用于应用的开发和运行阶段, 混合开发框架的实现大大提升了开发效率. 自移动应用开发平台上线以来, 先后对公司各业务中心的移动信息化建设进行了支持和服务工作, 包括 95598 掌上助手项目、电动汽车智能服务平台项目、普华移动办公 OA 系统等多个应用, 在国网的移动信息化建设中起到了举足轻重的作用.

4 结语

为了解决现有混合开发框架在便利性和性能方面的不足, 本文结合 HTML5 跨平台特性、打包机制、多窗口管理机制、原生插件开发技术等, 实现了一套自有的混合开发框架. 电动汽车等移动应用的实践证明, 该架构为混合应用的开发带来了便利的同时, 也提升了用户体验, 主要表现在如下几个方面:

(1)便利性: 开发人员不再需要同时使用 Native 框架 PhoneGap 和前端 JS 框架 JQuery Mobile 等相结合的方式进行开发, 只要采用本文设计的开发框架就可以完成混合的开发; 例如电动汽车应用(包括 Android 和 iOS 版本)为 2 个前端开发人员在两周之内完成, 现该应用在各应用市场以及 AppStore 上都可以下载到;

(2)共享性: 混合开发框架基于插件机制实现原生功能的调用, 原生插件只需要开发一次就可以用于任何项目, 达到了原生代码的共享功能; 同时, 框架中还提出了 widget 应用的概念, 跟原生插件类似, widget 应用是 Web 功能的共享; 这种共享显然对于企业资源的最大化利用起到非常重要的应用;

(3)性能优化: 本文提供了一套统一的窗口管理机制, 而该机制为原生实现, 窗口切换不再是单纯的靠 JS 和 HTML 来实现, 窗口切换动画同样采用原生动画实现, 这在很大程度上提升了混合应用的性能, 解决了混合应用用户体验不好的问题; 在电动汽车应用上线之前, 公司对其做了性能测试, 在内存消耗上虽然跟原生应用扔有差距, 但可以达到令人满意的效果.

(4)适配性: 由于引擎提供了统一的分辨率适配机制, 开发者可以为不同分辨率的设备写同一套代码, 这对于 Android 手机的适配起到重要的作用, 而且 iphone 上的应用即使在 ipad 上也不会出现布局混乱的问题. 公司对电动汽车应用做了不同 OS 版本和不同分辨率的适配, 效果都让人非常满意.

参考文献

1 武佳佳,王建忠.基于 HTML5 实现智能手机跨平台应用开发.软件导刊,2013,20(2):66–68.

2 Charland A, Leroux B. Mobile application development: Web vs. native. Communications of the ACM, 2011, 54(5): 49–53.

3 Hartmann G, Stead G, DeGani A. Cross-platform mobile development. Mobile Learning Environment, Cambridge, 2011.

4 董龙飞.关于 PhoneGap 的 7 件事.程序员,2012,(3):84–87.

5 张居彦.移动应用框架技术探析.电脑迷(上旬刊),2013,(3):20–20.

6 Vaughan-Nichols SJ. The mobile web comes of age. IEEE Computer, 2008, 41(11): 15–17.