

*Developing Applications in JavaScript
using the Palm Mojo™ Framework*

Palm webOS™

Rough Cuts

中英文对照版一吹友吧

www.treo8.com

Lila55 , CyberVsQ

O'REILLY®

Mitch Allen

译者序

本章节前半部分由 Lila55 翻译并发表于吹友吧的论坛 (www.Treo8.com)，后部分由 CyberVsQ 翻译。由于译者水平所限，难免有偏差和不详之处，请大家到吹友吧论坛斧正。

由于 webOS 还没发布，我们仅仅由此文章窥豹一斑，很多概念和原理无法了解，仅凭英文字面的意思生搬硬套，可能会对读者造成误导，因而本文以英语原文和中文对照的方式发布，如有不正确的地方以原文为准。

许多英文词汇在此文中有特别含义或特别所指，但是由于太新，没官方的译名，凭本人绞尽脑汁也无法想出更好的翻法，这个留待后面大家有更好的译名再进行纠正。

暂罗列如下：

Launcher 启动器 Framework 框架 Card 卡片 Notification 通知

Tagging 标签 Architecture 架构 native 原生

View 在 MVC 里面翻译为视图，在其他里面翻译为查看或者外观

Function 不明确为函数的地方暂译为功能，如比较显然所指为函数则直译为函数

Tap、Back 我们的理解这是操作的两个动作，前者类似于轻轻的点击，back 无解

Quick Launch bar 快速启动栏 Notification bar 通知栏

Activity 最近老在画活动图因而直译为活动，应该指的是一项有目的的操作行为

Dashboard 这个最难搞，看后文的图就知道是什么，但无法形象表达，暂译为指示板

headless application 无标题应用程序，指的应该是没卡片界面的程序

Stage、scene 分别翻译为舞台和场景，前者应该有更好的词汇可以替代

Picker、Viewer 采集器和查看器，都是一种组件，没见过只好直接翻译了

toggle button 切换按钮，应该是按一下凹陷，再按一下弹起的那种。

Catalog 编目，就是类似于 iPhone 采用的应用程序发布的列表

Overview of webOS 概述

Palm® webOS™ is Palm's next generation operating system. Designed around an in-credibly fast and beautiful user experience and optimized for the multi-tasking user, webOS integrates the power of a window-based operating system with the simplicity of a browser. Applications are built using standard web technologies and languages, but have access to device-based services and data.

Palm® webOS™是 Palm 公司新一代的操作系统。WebOS 将窗口操作系统的强大功能和浏览器应用的简便性融合在一起，提供了非常快速并且华丽的用户界面，而且还为多任务操作进行了优化。可以使用任何标准的 web 技术和语言撰写 WebOS 的应用程序，同时这并不会影响到 WebOS 应用程序对设备底层服务和本地数据的访问。

Palm webOS is designed to run on a variety of hardware with different screen sizes, resolutions and orientations, with or without keyboards and works best with a touch-panel though doesn't require one. Because the user interface and application model are built around a web browser, the range of suitable hardware platforms is quite wide, requiring only a CPU, some memory, a wireless data connection, a display, and a means for interacting with the UI and entering text.

Palm webOS 能够运行在许多设备上，这些设备在屏幕尺寸、分辨率以及用途上各不相同。Palm webOS 支持键盘操作，尽管如此，在配备触摸屏的设备上表现更好。因为 Palm webOS 的用户界面和应用程序都是基于浏览器应用的，所以，只要硬件设备包含一个 cpu、一些内存、支持无线数据连接、有显示设备、并且能够进行文本输入和用户界面操作，那么这个设备就能够使用 Palm webOS。

You can think of webOS applications as native applications, but built from the same standard HTML, CSS and JavaScript that you'd use to develop web applications. Palm has extended the standard web development environment through a JavaScript frame-work that gives standardized UI widgets, and access to selected device hardware and services.

尽管 Palm webOS 应用程序开发基于标准的 HTML, CSS and JavaScript（这些语言都是开发 web 应用程序必备的---注：web 应用程序是 web 应用程序，WebOS 应用程序是 WebOS 应用程序，不是一码事儿），我们仍然可以认为 WebOS 应用程序就是原生程序。Palm 已经通过一套完成的 JavaScript frame-work 对标准的 web 开发环境进行了扩展。这套 JavaScript frame-work 提供了标准化的用户界面组件，并且支持对设备硬件和服务的访问。

The user experience is optimized for launching and managing multiple applications at once. WebOS is designed around multi-tasking, and makes it utterly simple to run background applications, to switch between applications in a single step, and to easily handle interruptions and events without losing context.

用户可以在 WebOS 上同时启动并且管理多个应用程序，Palm 为此对 WebOS 进行了专门优

化。WebOS 的设计初衷就是为了更好地支持多任务，因此 WebOS 能够很容易地运行后台程序、能够在多个程序间一部切换，而且还能够非常轻松地在不丢失当前输入的情况下处理各种中断和系统事件。

You will build WebOS applications with common web development tools following typical design and implementation practices for Ajax applications. But your webOS applications are installed and run directly on the device, just as you are used to doing with native applications.

开发人员可以很容易地用常见的 web 开发工具进行开发，开发方式和开发 Ajax 应用程序一样。不同的是，WebOS 应用程序直接安装并且运行在设备上，这一点和以前的原生应用程序没有任何区别。

Application Model 应用程序模型

As shown in Figure 1-1, the original Palm OS has a typical native application model, as do many of the popular mobile operating systems. Under this model the application's data, logic and user interface are integrated within an executable installed on the native operating system, with direct access to the operating system's services and data.

如图1-1所示，最初的 PalmOS 应用程序和其他流行的移动操作系统一样都是原生程序。在这种情况下，程序的数据、逻辑以及用户界面都被整合在一个可执行应用程序中，然后安装在本地操作系统上，这样就能够直接访问操作系统服务和数据。

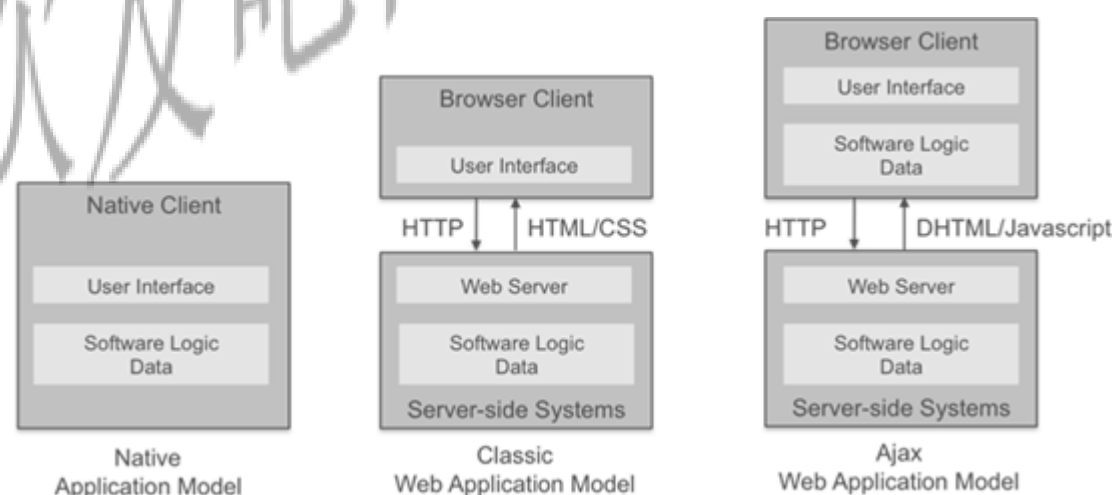


Figure 1-1. Native and Web Application Models

Classic web applications are basic HTML-based applications that submit an HTTP request to a web server after every user action, and wait for a response before displaying an updated HTML

page. More common in recent years are Ajax applications, which handle many user interactions directly and make web server requests asynchronously. As a result, Ajax applications are able to deliver a richer and more responsive user experience. Some of the best examples of this richer experience are the map applications, which enable users to pan and zoom while asynchronously retrieving needed tiles from the web server.

典型的 web 应用程序使用以 HTML 为基础的语言进行开发。在每一个用户动作之后，都会向一个 web 服务器提交 HTTP 请求，然后等待回应。收到回应之后，以 HTML 页面的形式对返回结果进行显示。最近几年更常见的是 Ajax 应用程序。Ajax 应用程序能够直接处理用户动作，并且能够以异步的方式向 web 服务器发出请求。因此，Ajax 就能够提供更丰富更快速的用户体验。地图程序就是个好例子。地图程序在持全景以及缩放功能的同时，也能够异步地从 web 服务器获取所需的数据。

Ajax applications have some significant advantages over embedded applications. They can be more easily deployed and updated through the same search and access techniques used for web pages. Developing web applications is far easier too; the simplicity of the languages and tools, particularly for building connected applications, allows developers and designers to be more productive. Connected applications, or applications that leverage dynamic data or web services, are becoming the predominant form for modern applications.

Ajax 应用程序相比于嵌入式程序有很大的优势。普通网页使用搜索访问技术，用同样的技术就能够很轻松地部署、升级 Ajax 应用程序。同样，开发 web 应用程序也极其简单：开发语言和开发工具很简单，这就大大提高了开发设计人员的效率，尤其是开发需要数据连接的程序时。因此，实时数据应用程序（Connected applications，不知道翻译的是否准确）以及支持动态数据和 web 服务的应用程序就成了主流。

The webOS application model combines the ease of development and maintenance of a web application with the deep integration available to native applications, significantly advancing the mobile user experience, while keeping application development simple.

然而，WebOS 应用程序，却进一步把 web 应用程序开发维护的简便性和本地应用程序的整体性结合在一起。极大地改善了移动用户体验，同时又保留了简便易行的开发方式。

Application Framework and OS 应用程序 框架和操作系统

Through Palm's application framework, applications can embed UI widgets with sophisticated editing, navigation and display features, enabling more sophisticated application user interfaces.

The framework also includes event handling, notification services and a multi-tasking model. Applications can run in the background, managing data, events and services behind the scenes while engaging the user when needed.

Palm 的应用程序框架在用户界面组件里集成了复杂编辑、导航以及一些显示特性，这样就能够开发出更为复杂的用户界面。这套应用程序构架还包括了事件处理、通知服务、以及多任务支持。应用程序在后台运行的同时能够管理数据、事件以及相关服务，用户需要的时候可以轻易切换到显示界面。

You can create and manage your own persistent data using HTML5 storage functions, and you can access data from some of webOS's core applications, such as Contacts and Calendar. You also have access to some basic system services, most of which are device-resident, such as Location services and Accelerometer data, along with some cloud services, such as XMPP messaging.

用户可以使用 HTML5 的存储函数对需要保存的数据进行管理，用户还能够访问 WebOS 核心应用程序的数据，比如联系人和日程的数据。用户也能够访问一些基本的系统服务，比如位置服务（应该和 gps 或者基站定位有关系）和重力感应器数据（这个应该是说 WebOS 支持重力感应应用，例如 iPhone 上的《古惑狼赛车》），这些系统服务都是常驻系统的，内嵌于设备的。除此之外，还支持一些晕服务，比如 XMPP 消息服务。

Architecturally, Palm webOS is an embedded Linux operating system that hosts a custom User Interface (UI) System Manager built on standard browser technology. The System Manager provides a full range of system user interface features including: navigation, application launching and lifecycle management, event management and notifications, system status, local and web searches, and rendering application HTML/ CSS/JavaScript code.

从整个构架上来说，Palm WebOS 是一种嵌入式 linux 系统。在这个系统上，可以通过标准的浏览器技术构建可定制的用户界面管理器。这套用户界面管理器提供了一整套系统用户界面的特性，包括：导航、应用程序启动和终止、事件管理和消息通知、系统状态、本地以及 web 搜索、渲染解释应用程序的 HTML/ CSS/JavaScript 代码。

You don't need to build a webOS application to make your web content accessible to webOS devices. Palm webOS has a separate browser application to handle standard web pages, and browser-based web applications. While it's expected that more and more web content and services will be delivered as webOS applications, there are millions of legacy websites and information that will continue to be presented in ways best viewed with a classic web browser. Palm webOS supports traditional web content very competitively.

用户不需要专门编写一个 WebOS 应用程序来浏览 web 内容。Palm WebOS 提供了一个单独的浏览器来对 web 内容和基于浏览器的 web 应用程序进行处理。尽管我们预计日后会有越来越多的 web 内容以 WebOS 应用程序的形式出现，我们仍然需要认识到，网上还有相当多的信息和网站采用老式代码和构架，对这些信息和站点的浏览而言，传统的浏览器仍然是最好的选择。但是，Palm WebOS 对传统内容的支持非常有竞争力。

Beyond the operating system, webOS includes a number of core applications: contacts, calendar, tasks, memos, phone, browser, email and messaging. Other applications are included in the initial release, such as a camera, photo viewer, audio/video player and map application, but the full application suite for a given webOS device will vary depending on the model and carrier configuration.

除了基础的操作系统，WebOS 还包括一整套核心应用程序：联系人、日程表、任务表、记事本、电话功能、浏览器、电子邮件以及短信。在最初发布的版本中，WebOS 还会提供额外的应用程序，比如照相机、照片浏览器、音视频播放器以及地图程序。但是，因为具体的硬件配置以及运营商要求不同，特定的 WebOS 设备包含的应用程序也不尽相同。

User Interface 用户界面

Palm webOS is designed for mobile, battery-operated devices with limited though variable screen sizes, and a touch-driven user interface. User interaction is centered on one application at a time, though applications, once launched, continue to run until closed even when moved out of the foreground view. There is a rich notification system enabling applications to subtly inform or directly engage the user, at the application's discretion.

Palm webOS 主要应用于手持的、由电池供电的、配备了触摸屏的设备。尽管这些设备的屏幕尺寸不同，但是都有一定限制。每次只能允许一个应用程序显示在屏幕中央，但是其他应用程序一旦启动，就会一直运行（即便没有被显示在屏幕中央位置）直到被关闭。Palm webOS 提供了丰富的通知功能，能够使得用户在使用当前程序的同时不会遗漏其他程序的通知。其他程序的通知可能是气泡式的，也能够直接切换到屏幕的中心位置。

Navigation 导航

Navigation is based upon a few simple gestures with optional extensions that create a rich vocabulary of commands to drive powerful navigation and editing features. To start with though, all you need to know is:

一些简单的手势操作就能够完成导航功能。这些手势操作还有一些可选的扩展功能。这样就能够用一套丰富的指令进行导航和编辑。一开始，用户需要掌握下一基本操作：

- tap (act on the indicated object). Commonly in a view that contains clusters or lists of items, tapping reveals information contained in an item. This can be thought of as an open function, which changes the nature or context of the view to be about the selected item exclusively.

Alternately, a tap will change an object's state such as setting a checkbox or selecting an object.

点击 (Tap) (对图标对象起作用): 在包含了一系列图标的界面中使用这个动作, 点击可以被用来启动应用程序。另外点击还能够改变当前应用程序的视图, 还能够用于选择一个对象或是设置复选框。

- back (the inverse of open). This feature looks like the opposite of a tap: the item compresses down to its summary in the containing context where it belongs. Typically, it reverses a view transition, as going from a child view to a parent view.

Back(open 的逆操作): 实际上是 tap 的逆操作, 被启动的程序会回到任务栏中(这里 summary 我暂时理解为类似于 windows 的任务栏), 并且整个显示退回到程序启动前的界面。

- scroll (flick and quick drags are used to scroll through lists and other views).

Scroll: 滚动, 用于在不同的列表和视图中进行切换。

Beyond this, you can learn to pan, zoom, drag & drop, switch applications, switch views, search, filter lists and launch applications. But to begin with, only these three gestures are needed to use a webOS device.

会了以上这些手势, 用户就能进一步掌握 pan (移动以获得全景)、缩放、拖放、切换程序、切换视图、搜索、过滤列表以及启动应用程序。但是一开始, 只需要掌握这三个动作就能够使用 webOS 设备。

Launcher 启动器

When you turn on a webOS device, the screen displays the selected wallpaper image with the status bar across the top of the screen and, hovering near the bottom, the Quick Launch bar. The Quick Launch bar is used to start up favorite applications or to bring up the Launcher for access to all applications on the device. From this view, a search can be initiated simply by typing the search string; searches can be performed on contacts, installed applications, or to start a web search. Figure 1-2 shows both the Quick Launch bar and the Launcher.

当用户打开一个 WebOS 设备时, 屏幕上会显示壁纸, 屏幕顶部会显示状态栏, 同时在屏幕底部漂浮着一个快速启动栏 (Quick Launch bar)。快速启动栏用来启动最常用的应用程序, 也能够切换出胖子们都很熟悉的启动器界面以便于看到设备上所有的应用程序。在启动器 (Launcher) 界面里, 简单的输入搜索字符串就能够在联系人以及其他应用程序里进行搜索, 还能够用这种方式进行 web 搜索。图1-2所示即为快速启动栏和启动器。

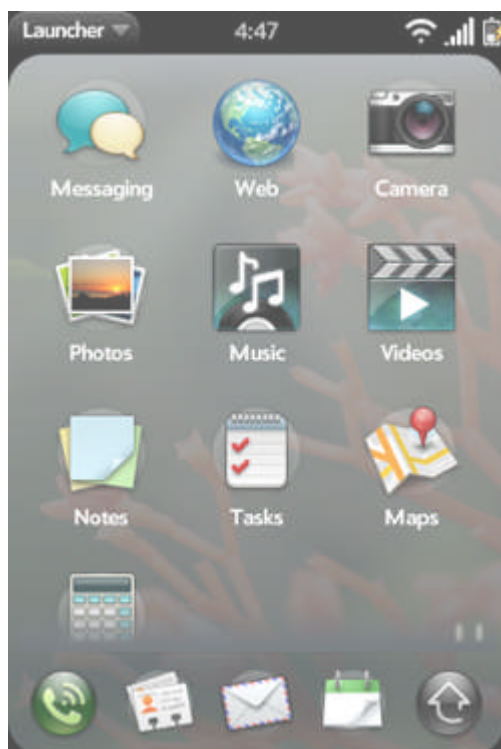


Figure 1-2. Quick Launch bar and Launcher

The launched application takes over the available screen becoming the foreground application; the application's view replaces the wallpaper image and the Quick Launch bar is dismissed. The status bar remains and is always visible except for full screen mode, which is available to applications such as the video player, or others that request it. This sequence is fluid and smooth, as you will see with all webOS transitions.

被启动的应用程序接着就会被显示在主要区域；当前应用程序界面替换了壁纸。并且快速启动栏（Quick Launch bar）也不再显示。除了某些应用程序的全屏显示（比如视频播放器），状态栏一直都位于屏幕顶端。整个过程非常流畅平滑，所有的 webOS 界面切换都是这样的。

Card View 卡片式界面

Figure 1-3 shows an application's main view, in this case the email application's folder view. The application view includes UI elements that make up the basic email application, in this case the inbox view displays specific folders, which when selected will open a new card with a detail view of the messages contained within the selected folder. At the bottom, the floating icons that you see are menu items. A tap to the menu icons will typically reveal another view associated with the menu action, a sub-menu or a dialog.

图1-3显示了邮件程序的主界面。在这个界面里，一些用户界面组件构成了这个基本的电子

邮件程序。其中，收件箱界面显示特定的文件夹，选定这个文件夹时，就会切换到一个新界面，这个界面显示的是当前文件夹里邮件的一些详细内容。这时，在底部漂浮的图标就是菜单按钮。点击这种漂浮图标将会打开菜单、显示子菜单或者打开对话框。

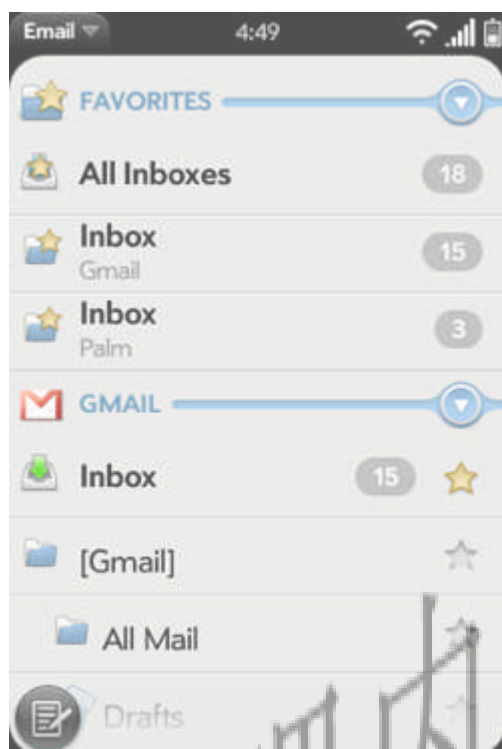


Figure 1-3. Email Application

But running one application at a time, or performing one activity at a time, can be terribly restrictive, and inefficient. Palm webOS was designed to make it easy to work on more than one thing at a time. Simply pressing the Center button brings up a new view, the Card view, an example of which is shown in Figure 1-4. From the Card view, you can switch to another activity simply by scrolling to and tapping the card representing the activity. Alternately, you can launch another application from the Quick Launch bar.

但是，一次运行一个程序，或者执行一个任务，显然无法发挥 WebOS 的能力。WebOS 设计的初衷就是能够同时运行多个程序，并在这些程序间切换，协调这些程序之间的合作。只需要按一下中键（Center button）整个界面就会变成卡片式界面，如图1-4所示。在卡片式界面里，用户可以使用点击（tapping）和卷动（scrolling）手势轻松地切换应用程序和显示界面。或者，也可以直接从 Quick Launch bar 里启动新的应用程序。

The Card view was inspired by the way one handles a deck of cards. Cards can be fanned out to see what card is where. Within a deck of cards, any single card can be selected or removed with a simple gesture, or moved to a new location by slipping it between adjacent cards. The webOS Card view can be manipulated in similar ways by scrolling through the cards, selecting and flicking cards off the top to remove them or selecting and dragging a card to a new location.

我们从平时玩纸牌的动作中获得灵感，由此开发了卡片式界面。用户可以摊开卡片，以便于看到哪张卡片在哪里。现实生活中，一副卡片里，用户可以通过简单的动作对卡片进行选择或者移动，也可以把一张卡片放在两个相邻卡片之间。WebOS 的卡片界面操作方式与此类似，用户可以在卡片间卷动、选择并且单击卡片以便于移开、或者选择并且拖拽卡片到一个新的位置。

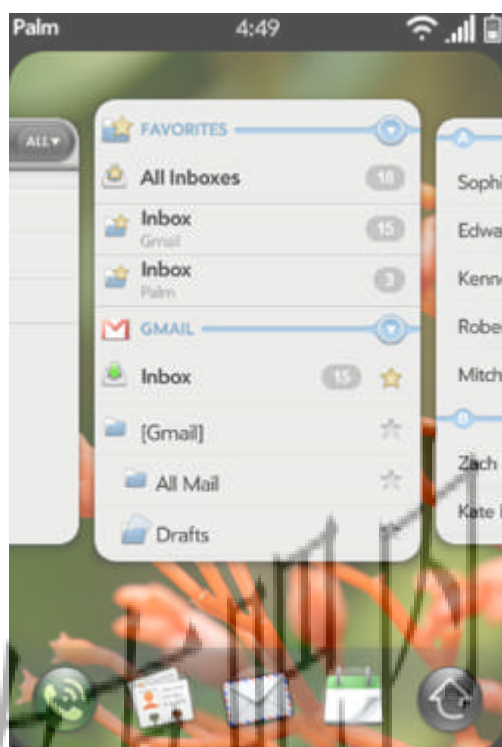


Figure 1-4. Card view with Email app and other apps

We've introduced the term activity, which needs further explanation. Often by design, you will work on one activity at a time with many applications, but with some applications it is more natural to work on several activities in parallel. A common email activity is writing a new email, but in the middle of writing that email, you may want to return to the inbox to look up some information in another email or perhaps read an urgent mail that has just arrived.

尽管前面已经提到活动(activity)这个概念，这里仍然有必要对这个概念做进一步的解释。通常，用户能够同时运行很多程序，但是每次只能执行一个活动，但是在某些情况下，同时进行多个活动更加自然合理。举个例子，一个常见的邮件活动就是写一封新的电子邮件，但是在写这封邮件的时候，用户也许需要返回到收件箱查看其它邮件里的信息，或者正写着当前邮件，又来了一封紧急邮件。

With a webOS device, the draft email has its own card separate from the email inbox card. In fact, you can have as many draft emails, each in their own card, as you need; each is considered a

separate activity and independently accessible. Switching between emails is as simple as switching between applications and your data is safe, as it is always saved. Figure 1-5 shows the Card view with the Email application's inbox card and a draft email compose card.

在 webOS 设备上，邮件程序有独立的草稿卡片。事实上，只要用户需要，就可以同时撰写多个邮件草稿，每个邮件草稿都对应一个卡片。每个卡片都被看作是分别的独立的活动（activity），并且能够分别查看并编辑不同的卡片。在不同的邮件间进行切换与在不同的应用程序间进行切换一样简单。用户不用担心自己的数据会丢失（到底还是有一些 palm 的精髓---不需要“保存”，就像我以前写过的，“palm 就像是一堆用不完的白纸”）。图1-5显示的就是电子邮件程序的卡片界面。

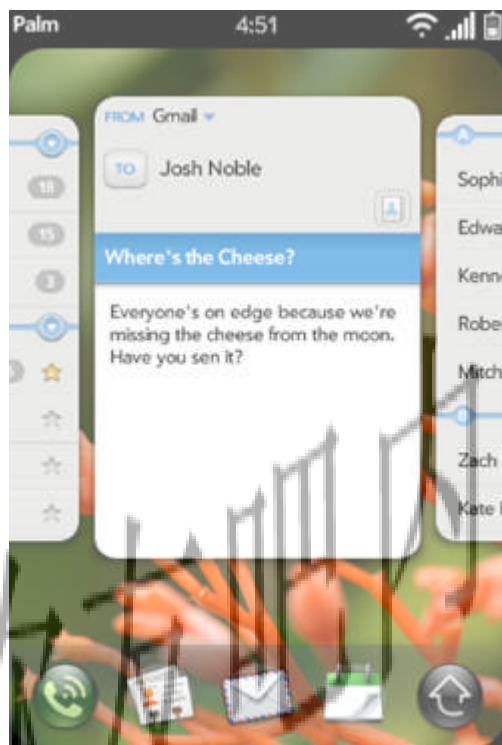


Figure 1-5. Card view with Email Application and New Email

Notifications and the Dashboard 事件通知和指示板

What happens to the current foreground application when you switch to a new appli-cation? The previous application is not closed but continues to run as a background application. Background applications can get events, read and write data, access services, repaint themselves and are generally not restricted other than to run at a lower priority than the foreground application.

如果用户切换到一个新的程序，在屏幕的主界面上会发生什么？新程序之前的那个程序并没有被关闭，只是转为后台运行。转为后台运行的程序照样可以获得系统事件、读写数据、访问系统服务、改变自己的界面，只不过比当前显示在主界面的程序而言，运行优先级低了一点。

To enable background applications to communicate with the user, Palm provides a notification

system with two types of notifications:

- Popup. Non-modal dialogs which are of fixed height and include at least one button to dismiss the dialog
- Banner. Non-modal icon and single non-styled string of text

Palm 提供了一个通知系统来帮助用户和后台程序进行沟通，这个通知系统有两种类型的通知：

- 气泡式通知：一个固定高度的非模式对话框，包括至少一个关闭按钮。（此概念与模式对话框 modal dialogs 相对，指在不关闭当前对话框的情况下仍然可以进行其他任务，而模式对话框则需要先行关闭当前对话框才能够进行其他的任务。）
- 横幅式通知：非模式化的图标，外加一行无风格的文本信息（这里 non-styled 应该是指对字体没有经过色彩或者其他形式的渲染）

Popup notifications are disruptive, appropriate for incoming phone calls, calendar alarms, navigation notifications and other time sensitive or urgent notifications. Users are forced to take action with pop-ups or explicitly clear them but since they are not modal, users are not required to respond immediately.

气泡式通知具有抢占性。用于来电、日程提醒、导航通知以及其他对时间敏感的或者紧急的通知事件。当出现气泡式通知时，用户必须对气泡式通知采取相关动作，或者直接清除它们。但是，因为气泡式通知是非模式的通知，所以用户不一定要立即对他们做出回应。

Banner notifications are displayed in a slow crawl along the bottom of the screen within the Notification bar, which sits just below the application window in what is called negative space since it is outside of the card's window. After being displayed, banner notifications can selectively leave a summary icon in the Notification bar as a reminder to the user. Figure 1-6 shows an example of a banner notification and the summary icons are shown in Figure 1-7 indicating that the music player is active and that there is an upcoming calendar event and new messages.

横幅式通知显示在屏幕底部的通知栏(Notification bar)，并且会缓慢的横向移动（有点像LED 电子横幅）。通知栏位置就在应用程序窗口正下方，因为这个位置在卡片窗口之外，所以被称作(negative space)。在显示了横幅式通知之后，用户可以选择是否在通知栏里保留这个横幅式通知的概要图标(summary icon)，以便于随后提醒用户还有未处理的通知。图1-6显示了一个横幅式通知，图1-7显示的概要图标告诉用户后台还运行着一个音乐播放器、有一个即将出现的日程提醒、并且还有一条新短信。

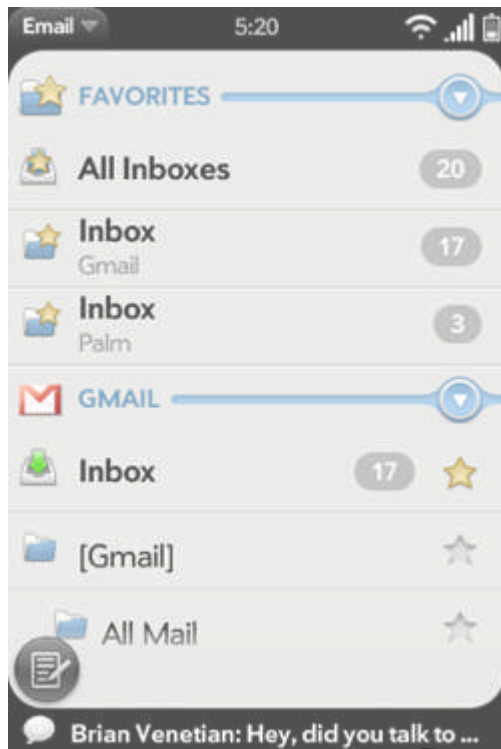


Figure 1-6. Banner Notification

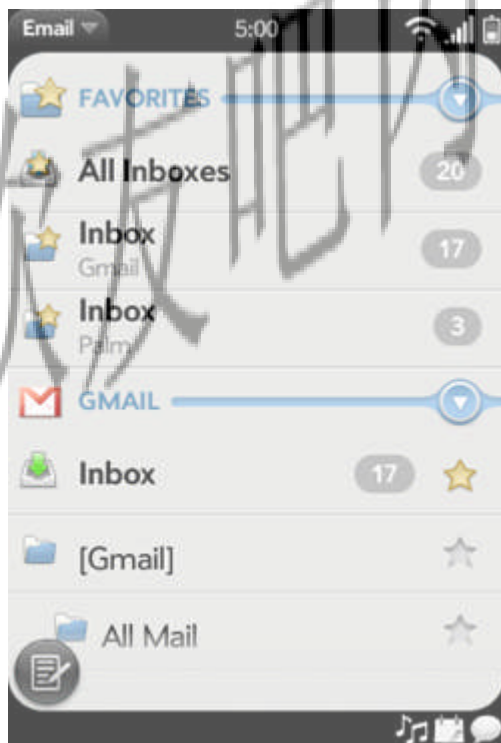


Figure 1-7. Notification Icons

changes in information without disrupting your current activities. It may help to think of them as an event-driven model for viewing and managing your world, while the Card view provides you with task-oriented navigation tools. The combination gives you a few powerful tools from which you can quickly track and access what you need when you need it.

通知栏(Notification Bar)和指示板能够很好地管理系统事件和各种中断,使得用户可以及时掌握信息的变化,却不用停下手头正在干的活儿。如果不好理解,那么可以把这个通知系统看成是一个由事件驱动的系统,而相应的,卡片界面则可以被看作是一个由任务驱动的系统。这两者结合在一起,可以极大的提高用户的工作效率,可以随时获取用户所需要的信息,而又不需停止当前任务。

Headless applications are those that can be completely served through the dashboard, as their entire purpose is to monitor and present information. For example, a weather application could display the current weather for a targeted location in a dashboard without having a card view at all.

无标题应用程序(Headless applications)指的是那些不需要主界面显示,只需要 dashboard 就能够完全控制的应用程序。因为这种程序的唯一目的就是监控并且显示信息。比如,一个天气预报程序的功能也不过是现实某个指定地点的天气情况,完全不需要占用卡片界面,显示在 dashboard 就足够了。

You will tend to use the Card view to switch between tasks, launch applications and otherwise perform activities. Dashboard is used to monitor your world, to see what's changed or what events have taken place, which will often drive new activities.

用户会习惯于使用卡片界面来切换程序、启动程序或者用来执行某些任务。Dashboard 则能够像个贴心助手那样帮助用户监控信息变化,看看什么即将发生,或者已经发生了什么,然后再由用户决定是否需要开展新的任务。

User Interface Principles 用户界面的一些原则

There are some foundational principles or values that support the overall webOS user experience; application designers can exploit these same principles to more deeply in-tegrate the application into the overall device experience and enhance the user's expe-rience. Developers can rely on the framework to provide most of what is required at an implementation level, but the application design should anticipate these needs.

WebOS 的用户界面开发需要遵循一些基本的原则。开发人员按照这些原则进行开发将能够把应用程序的体验和对设备整体的体验全面地结合在一起,使用户察觉不到程序间的差异,以及程序和整个系统之间的差异(PalmOS 就遵循这样的原则,每个程序看起来风格都很统一)应用程序的开发可以依赖 SDK 实现所需功能,但是设计程序的时候一定要考虑到这些基本原则。

Here are the key principles to keep in mind while designing your application:

下列就是在程序设计阶段，开发人员需要牢记在心的东西：

- Physical metaphors are reinforced through direct interaction with application objects and features, instant response to actions, followed by smooth display and object transitions with physics-based scrolling and other movement. For example, objects are deleted by flicking off screen and editing is in place without auxiliary dialogs or scenes.

- 要尽量真实，模仿现实情况，秉承 palm 的一贯特色。通过直接和应用程序对象互动能够增强真实性。程序对动作的反应要足够快，要能够流畅显示操作对象的变化，这些只需要用户做出最自然的动作即可。比如，在屏幕上轻轻拨动就能够删除一些东西，或者不需要繁琐的对话框就能够直接编辑文本。

- Maintain a sense of place with repeatable actions, reversible actions, stable object placement and visual transitions taking the user from one place to the next.

- 要有场景的概念（sense of place），可重复、可逆的操作、程序对象在屏幕上相对固定的位置，以及视觉上的变化都应该能够把用户从一个场景带到另一个场景里。

- Always display up-to-date data, which requires both pushing and pulling the latest data onto the device so that the user is never looking at stale data when more recent data is available. But this also means managing on-device caches so that when the device is out of coverage or otherwise off-line, the user has access to the last data received.

要保证所显示的数据是最新的。这就要求应用程序获取最新的数据，并且及时显示在设备上。这样，用户就能够及时掌握信息变化。但是，这也意味着应用程序要和设备的 cache 打交道。因为，当设备不在信号区或者离线，用户应当能够获取设备在离线之前收到的最新信息。

- Palm webOS is fast and simple to use; all features should be designed for instant response, easy for novices to learn while efficient for experienced users.

- Palm WebOS 用起来应该很快，但是却很简单。所有的动作都要立即响应。对新手应该是很容易上手的，而对老鸟，则应该是非常高效的。（到底还是有 palm 的血统）

- Minimize the steps for all common functions; put frequently executed commands on the screen, the next most frequent under the menus. Avoid preferences and settings where possible and where not, keep them minimal.

- 省去一切不该有的步骤！把最常用的命令直接显示在屏幕上，次之的放在菜单里。尽可能

的避免诸如“特性”“设置”之类的菜单，就算需要这一类东西，也要尽量简洁明了。

- Don't block the user; don't use a modal control when the same function can be carried out non-modally.

- 不要妨碍用户！在不需要模式化的地方，就不要用模式化！（比如尽量不要“打开”、“保存”之类的对话框或按钮，还是那句话，palm 在手里，就好象一堆用不完的白纸！）

- Be consistent; help the user learn new tasks and features by leveraging what they have already learned.

- 保持一致。要利用用户已经掌握的的东西来帮助用户学会新的任务，掌握新的程序特点。

Palm applications have always been built around a direct interaction model, where the user touches the screen to select, navigate, and edit. Palm webOS applications have a significantly expanded vocabulary for interaction, but they start at the same place. Your application design should be strongly centered on direct interaction, with clear and distinguishable targets. The platform will provide physical metaphors through display and navigation, but applications need to extend the metaphor with instantaneous re-sponse to user actions, to smoothly transitioning display changes, and object transi-tions.

Palm 应用程序一直以来都采用直接互动模式。在这种模式下，用户只需要在屏幕上触摸就能够进行选择、导航、编辑等动作。Palm webOS 应用程序比起 Palm 应用程序而言，支持更多的互动方式。但是 Palm webOS 应用程序的基本操作都一样。开发人员设计程序时，一定要围绕着“直接互动”这一模式，要让用户觉得直观明了。Palm webOS 通过自己的导航和显示功能，真实地模拟了实际情况。但是，应用程序则需要在这个基础上进一步提高对用户动作的响应速度，进一步模拟真实情况（要把 Palm 当成手头上一堆用不完的白纸！不需要打开，不需要保存，更不需要看见某种恶心的沙漏式的东西！）。

You can find a lot more on the user interface guidelines and design information in the Palm webOS SDK under the Design Guide. We'll touch on the principles and reference standard style guidelines in the next few chapters, but will not be covering this topic in depth.

在 Palm WebOS SDK 的设计指南（Design Guide）里，开发人员能够找到更多的有关于程序设计和用户界面的信息。在随后的章节里，我们将会进一步介绍这些原则，还会介绍一些标准设计的指导方针，但是，不会谈的太深入。

Mojo Application Framework 应用程序框架

A webOS application is similar to a web application based on standard HTML, CSS, and JavaScript, but the application lifecycle is different. Applications are run within the UI System Manager, an application runtime built on standard browser technology, to render the display, assist

with events, and handle JavaScript.

webOS 应用程序与基于标准 HTML, CSS 和 JavaScript 的 Web 程序类似, 但是应用程序的生命周期不一样, 应用程序运行在用户界面系统管理器(UI System Manager)里面, 这是一个用标准浏览器技术构建的应用程序运行库, 用于显示渲染, 并辅与事件和处理脚本。

The webOS APIs are delivered as a JavaScript framework, called Mojo, which supports common application-level functions, UI widgets, access to built-in applications and their data, and native services. To build full-featured webOS applications, many developers will also leverage HTML5 features such as video/audio tagging and database functions. Although not formally part of the framework, the Prototype JavaScript framework is bundled with Mojo to assist with registering for events and DOM handling among many other great features.

webOS 应用程序接口作为名叫 Mojo 的 javascript 框架来发布, 它能支持一般应用程序级别的功能, 比如界面组件, 访问内置的应用程序和他们的数据及原生的服务。为了构建全功能的 webOS 应用程序, 开发者也要深入了解 HTML5 的特征, 比如视频/音频标签 (tagging) 和数据库功能。框架的原型捆绑了 Mojo 来辅助处理许多其他强大功能之间的事件注册和 DOM 处理, 但这些不是作为框架的正式功能。

The framework provides a specific structure for applications to follow based on the Model-View-Controller (MVC) architectural pattern. This allows for better separation of business logic, data, and presentation. Following the conventions reduces complexity; each component of an application has a defined format and location that the framework knows how to handle by default. You will get a more extensive overview of Mojo in Chapter 2, and details on widgets, services and styles starting in Chapter 3. For now, you should know that the framework includes:

- Application structure, such as controllers, views, models, events, storage, notifications, logging and asserts;
- UI widgets, including simple single-function widgets, complex multi-function widgets and integrated media viewers;
- Services, including access to application data and cross-app launching, storage services, location services, cloud services, and accelerometer data;

框架给应用程序提供了一个特殊的结构以适应 MVC 的架构模式。可以更好的分离业务逻辑、数据和表现层。根据惯例, 为了降低复杂度, 应用程序的每个部件都有一个定义好的格式和位置, 这样框架可以知道缺省怎么去做处理。你将在第二章中获得对 Mojo 的更多信息和页面构件的更详细信息, 而服务和风格将在第三章开始讲述。现在, 你应该知道框架包括以下方面:

- 应用程序结构, 就如控制器、视图、模型、事件、存储、通知, 日志和维护。
- 用户界面组件, 包括简单的单一功能构件, 复杂的多功能构件和集成的多媒体查看器。

- 服务，包括访问应用程序数据和交叉应用执行，存储服务，位置服务，云服务和重力感应服务。

Anatomy of a webOS Application 剖析 webOS 应用程序

Outside of the built-in applications, webOS applications are deployed over the web. They can be found in Palm's App Catalog, an application distribution service, built into all webOS devices and available to all registered developers. The basic lifecycle stages are illustrated in Figure 1-9.

除了内置的应用程序，其他 webOS 程序通过 web 方式来部署。他们可以在 Palm 的应用程序编目里面找到，webOS 设备都内建了提供给所有的注册开发人员的应用程序分发服务，基本的生命周期如图1-9。

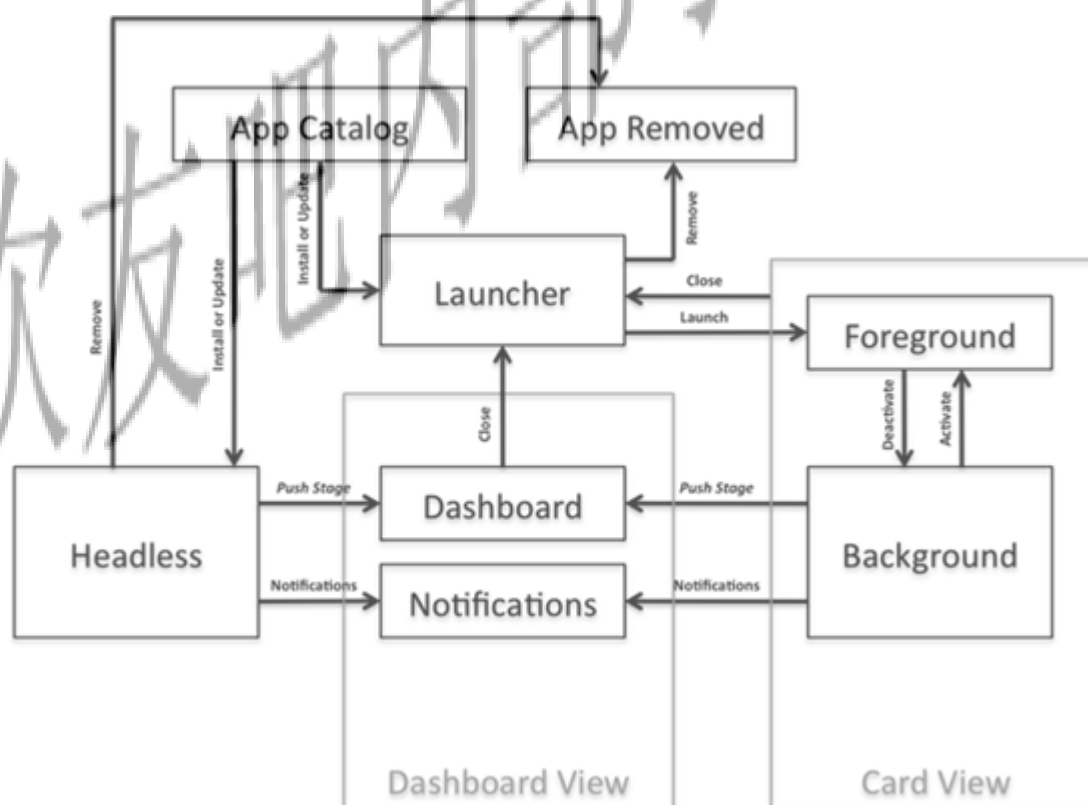


Figure 1-9. Application Stages

Downloading an application to the device initiates installation of the app provided that it has been

validly signed. After installation the application will appear in the Launcher. If it is a headless application, then a card is not required and instead the application can utilize just a dashboard and communicate to the user through notifications. Headless applications typically include a simple card based preferences scene to initiate the application and configure its settings. Note that headless applications require at least one visible stage at all times (either a card, dashboard or alert) to not be shut down.

应用程序（假如已经被有效的签名）下载到设备并开始安装。安装以后，应用程序将在启动器（Launcher）里面显示。如果是一个无标题应用程序（headless application），那么卡片不是必须的，应用程序可以只显示在指示板（dashboard），用通知提醒的方式和用户进行沟通。典型的无标题程序为简单的卡式参数选择界面用于初始化或者设置应用程序。需要注意，无标题程序需要至少在一个舞台是一直可见不被关闭的（无论是通过卡片、指示板还是警告）（我的理解是因为无标题程序不在那个切换来切换去的卡片里面出现，因而应该有一个地方能够去激活或者关闭，避免成为没法关闭的后台程序）

Other applications are launched from the launcher into the foreground and may be switched between foreground and background by the user. Each of these state changes (launch, deactivate, activate, close) is indicated by one or more events. Applications are able to post notifications and optionally maintain a dashboard while in the background.

其他程序（除了上面说的无标题程序）可以在启动器里面执行，由用户进行前后台进行切换。每个状态（执行、取消激活、激活、关闭）之间切换都会有一个或多个事件激发。应用程序可以发出提醒，如果在后台也可选的改变指示板。

Applications are updated periodically by the system. If running, the application is closed, the new version installed, and then it's launched. There isn't an update event so the app needs to reconcile changes after installation, including data migration or other compatibility needs.

系统会定期去更新应用程序。如果应用程序在运行，安装新版本时候被关闭，然后再运行。由于没有更新事件触发，所以应用程序需要在安装之后重新协调，包括数据迁移或者其他需要的兼容性处理。

The user can opt to remove an application and its data from the device. When the user attempts to delete an application, the system will stop the application if needed and remove its components from the device. This includes removing it from the launcher and any local application data, plus any data added to the Palm application databases such as Contacts or Calendar data.

用户可选择从设备上删除应用程序和数据。当用户试图删除一个应用程序，系统会在需要的时候停止该程序，然后从设备移除程序组件。包括从启动器上移除和所有的本地应用程序数据，还有 Palm 应用程序数据库的任何数据，例如联系人或者日历表数据。

Stages and Scenes 舞台和场景

Palm's user experience architecture provides for a greater degree of application scope than is

normally considered in a web application. To support this and specific functions of the framework, Palm has introduced a structure for webOS applications built around stages and scenes.

与普通 Web 程序相比, Palm 的用户体验架构提供了一个超乎应用程序级别的视域, 为了支持这些和特殊的框架功能, Palm 推出了围绕着舞台(stages)和场景(scenes)来构建的 webOS 应用程序结构。

A stage is a declarative HTML structure similar to a conventional HTML window or browser tab. Palm webOS applications can have one or more stages, but typically the primary stage will correspond to the application's card. Other stages might include a dashboard, or other cards associated with different activities within the application. You should refer back to the earlier example of the Email application where the main card held the Email inbox and another card held a draft Email. Each email card is a separate stage, but part of the same application.

舞台(stage)是一个定义的 HTML 结构, 类似于一个一般的 HTML 窗口或者浏览器标签。webOS 程序可以有一个或多个舞台, 但是一般主舞台相当于应用程序的卡片。其他舞台包括指示板(dashboard)或者应用程序里面其他活动相关的卡片。参考前文的 Email 应用程序例子, 主卡片包含收件箱, 其他卡片电子包含邮件草稿, 每个邮件卡片是一个独立的舞台, 但是是同一个程序的一部分。

Scenes are mutually exclusive views of the application within a Stage. Most applications will provide a number of different kinds of scenes within the same stage, but some very simple applications (such as Calculator) will have just a single scene. An application must have at least one scene, supported by a controller, a JavaScript object referred to as a scene assistant, and a scene view, a segment of HTML representing the layout of the scene

场景是应用程序在一个舞台中相互独立的外观。大多数程序将在同一个舞台提供许多不同类型的场景, 但是有些非常简单的应用程序(比如计算器)将只有一个场景。一个应用程序必须至少有一个场景, 一个类似于场景助理(assistant)的 JavaScript 对象和一个场景外观(view), 一段描述场景布局的 HTML 代码。

Most applications will have multiple scenes. You will need to specifically activate (or push) the current scene into the view and pop a scene when it's no longer needed. Typically, a new scene is pushed after a user action, such as a tap on a UI widget and an old scene is popped when the user gestures back.

大多数应用程序会有多个场景。你需要明确的激活（或者推动）当前的场景可见并弹开（Pop）不再需要的场景。典型的情况是一个新的场景在用户操作之后激活，就象在用户界面组件上点击一下，然后一个旧的场景被弹到背后。

As the terms imply, scenes are managed like a stack with new scenes pushed onto and off of the stack with the last scene on the stack visible in the view. Mojo manages the scene stack but you will need to direct the action through provided functions and respond to UI events that trigger scene transitions. Mojo has a number of stageController functions specifically designed to assist you, which you can find in detail in Chapter 2, Application Basics, and Chapter 3, UI Widgets.

按词索意，场景就象堆栈一样管理，一个新的场景推入，最后一个场景被推出来可见。Mojo 管理场景堆栈，但是你必须提供函数和用户界面的响应来触发指定操作的场景变迁。在第二章 应用程序基础和第三章 用户界面组件中会详细的说明 Mojo 设计的大量舞台控制器（stageController）函数来协助完成这些。

Application Lifecycle 应用程序生命周期

Palm webOS applications are required to use directory and file structure conventions to enable the framework to run the applications without complex configuration files. At the top level the application must have an appinfo.json object, providing the framework with the essential information needed to install and load the app. In addition, all applications will have an index.html file, an icon.png for application's Launcher icon, and an app folder, which provides a directory structure for assistants and views.

webOS 的应用程序要使用约定的目录和文件结构，这样框架不需要复杂的配置文件就可以运行应用程序。在应用程序的顶层必须有一个 appinfo.json 对象，给框架提供安装和加载程序的重要信息。所有应用程序都有一个 index.html 文件，一个 icon.png 文件提供启动器图标和一个 app 的文件夹来提供一个目录结构提供协助（assistants）和视图（views）。

By convention, if the app has images, other javascript or application-specific CSS, then these should be contained in folders named images, javascripts, and stylesheets respectively. This is not required but makes it simpler to understand the application's structure.

一般来说，如果应用程序有图片，其他脚本或者应用程序特定的 CSS，那么这些要分别放在 images, javascripts 和 stylesheets 的文件夹里面。这些让应用程序的结构易于理解但不是必须的。

Launching a webOS application starts with loading the index.html file and any referenced stylesheets and javascript files, as would be done with any web application or web page. However, the framework intervenes after the loading operations and invokes the stage and scene assistants to perform the application's setup functions and to activate the first scene. From this point, the application would be driven either by user actions or dynamic data.

webOS 应用程序的执行与一般 web 应用程序或者 web 页面一样从加载 index.html 和其他相关 stylesheets 和脚本开始，在加载完成之后，框架开始干预并调用舞台和场景来协助完成应用程序设置功能并激活第一个场景。从这个时候开始，应用程序将被用户操作或者动态数据驱动运行。

Significantly, this organizational model makes it possible for you to build an application that will manage multiple activities, that will be in different states (active, monitoring and background) at the same time.

值得注意的是这个结构模型使得用户可能建立一个应用程序管理多重的活动，同一个时间将处于不同的状态（激活、监控和背景）。

Applications can range from the simple to the complex:

- Single scene apps, such as a Calculator, which the user can launch, interact with and then set aside or close;
- Headless apps, such as traffic alert application that only prompts with notifications when there is a traffic event and whose settings are controlled by its dashboard;
- Connected apps like a social-networking app, which provides a card for interaction or viewing and a dashboard giving status;
- Complex multi-stage apps like Email, which can have an Inbox card, one or more Compose cards, along with a dashboard showing email status. When all the cards are closed, Email will run headless to continue to sync email and post notifications as new emails arrive.

应用程序可以由简单到复杂：

- 单一场景应用程序，就象计算器，用户可以执行，交换和搁置或者关闭；
- 无标题应用程序，就象交通（traffic）预警应用程序，只要当发生交通事件的时候弹出通知消息，而设置由指示板（dashboard）来控制；
- 联网应用程序，如一个社会网络应用，由一个卡片来交互或者查看，在指示板显示状态；
- 复杂的多状态应用，就象 Email，有收件箱卡片，一个或多个组合的卡片，也由指示板显示电子邮件状态。当所有的卡片关闭以后，Email 将无标题的继续运行来同步邮件，

当收到新邮件的时候弹出通知。

Events 事件

Palm webOS supports the standard DOM Level 2 event model. For DOM events, you can use conventional techniques to listen for any of the supported events and assign event handlers in either your HTML or JavaScript code.

webOS 支持标准的 DOM 两层事件模型。根据 DOM 事件，可以使用传统的技术来监听任何支持的事件并且在 HTML 或者 JavaScript 代码中指定事件的处理器。

The UI Widgets have a number of custom events, which are covered in more detail in Chapter 3. For these events you will need to use custom event functions provided within the framework. Mojo events works within the DOM event model but includes support for listening to and generating custom Mojo event types and is more strict with parameters; Mojo checks parameters to confirm that they are properly defined and typed.

在第三章中会全面的讲解用户界面组件的大量自定义事件。你应该用 Mojo 框架、DOM 事件模型提供的自定义的事件功能来处理这些事件，对于支持监听和生成自定义 Mojo 事件类型的参数必须严格定义，Mojo 检查参数来确认定义和类型是否正确。

The webOS Service functions work a bit differently, with registered callbacks instead of DOM-style events, and are covered starting in Chapter 7. The event-driven model isn't conventional to web development, but has been part of modern OS application design and derives from that.

webOS 的服务功能有一点特别，它用注册回调的方式来替代 DOM 类型的事件，这方面的内容在第7章讲述。事件驱动模型不是传统的 Web 开发，但是已经成为现代操作系统应用程序设计和起源的一部分。

Storage 存储

Mojo supports the HTML5 database functions directly and provides high-level functions to support simple Create, Read, Update or Delete (CRUD) operations on local databases. Through these Mojo Depot functions, you can create a local database and add, delete or retrieve records individually or as a set. It's expected that you'd use databases for storage of application preferences, or cache data for faster access on application launch or for use when the device is disconnected.

Mojo 直接支持 HTML5 的数据库功能并提供高级的功能来支持在本地数据库上的简单新建、读取、更新或者删除操作。通过这些 Mojo 新功能，你能建立一个本地数据库，添加，删除，个别或者批量的读取记录。数据库是应用程序用于存储，或者应用程序开始执行、设备断开连接时候缓存数据的优先选择，以便更快的数据访问。

UI Widgets 用户界面组件

Supporting webOS's user interface are UI Widgets and a set of standard styles for use with the widgets and within your scenes. Mojo defines default styles for scenes and for each of the widgets. You get the styles simply by declaring and using the widgets, and you can also override the styles either collectively or individually with custom CSS.

用户界面组件和一系列标准的风格（应用于组件和场景）构成 webOS 的用户界面。Mojo 定义了场景里每个组件缺省的风格，你可以通过定义和使用组件就简单的获得这些风格。你也可以用自定义的 CSS 来全面或有选择的覆盖这些风格。

The List is the most important widget in the framework. The webOS user experience was designed around a fast and powerful list widget, binding lists to dynamic data sources with instantaneous filtering and embedding objects within lists including other widgets, including other lists, icons and images.

列表的是框架中最重要的组件。webOS 的用户体验围绕着快速和强大的列表组件来设计，列表可以绑定动态数据源，提供快速的过滤，能够嵌入其他组件对象，包括其他列表、图标和图像。

There are Simple Widgets, including buttons, checkboxes, sliders, indicators, and containers. The Text Field widget includes text entry and editing functions, including selection, cut/copy/paste, and text filtering. A Text Field can be used singly or in groups or in conjunction with a List widget.

简单的组件包括按钮、复选框、滑动条、指示器和容器。文本输入框有文本输入和编辑功能，包括选择、剪切/复制/粘贴和文本过滤。一个文本框可以单独使用，也可以成组使用或者用列表组件来组合使用。

Menu widgets can be used within specified areas on the screen; at the top and bottoms are the View and Command menus and they are completely under your control. The App Menu is handled

by the system, but you can provide functions to service the Help and Preferences items or add custom items to the menu. Each of the various menu types is shown in Figure 1-10.

菜单组件使用时处于屏幕的一个特殊位置，外观和命令菜单在界面的顶端或者底部，这些完全在你控制之下。应用程序菜单由系统处理，但你可以提供函数来支持帮助和可选选项或者增加自定义的菜单选项。每种菜单类型如图1-10所示。

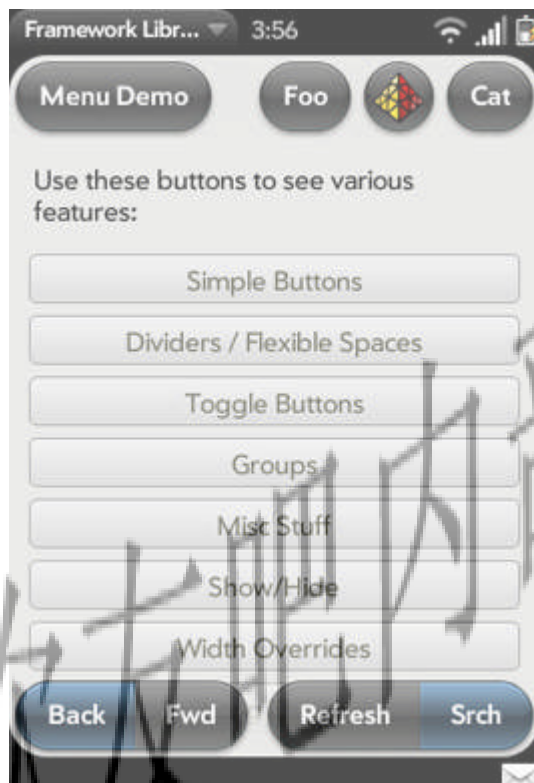


Figure 1-10. Application Menu Types

For Notifications, you can choose from a Popup Notification or a Banner Notification, both of which post notifications for applications in the notification bar.

通知功能可以选择弹出式通知或者横幅式通知，这两种组件都在通知栏发布应用程序的通知信息。

Pickers and Viewers are more complex widgets. Pickers are for browsing and filtering files or contacts, or for selecting addresses, dates or times. If you want to play or view content within your application, such as audio, pictures, video or web content, then you would include the appropriate

viewer.

采集器和查看器是较复杂的组件。采集器用于浏览和过滤文件或者通讯录，还可以用来选择地址，日期或时间。如果你想播放或者查看应用程序的内容，例如音频，图片，录像或者 web 内容，那么你要用恰当的查看器。

Using Widgets 使用组件

A widget is declared within your HTML as an empty div with an x-mojo-element attribute. For example, the following declares a Toggle Button widget:

组件在 HTML 中进行定义，就如一个空的带 x-mojo-element 属性的 div，例如下面定义了一个切换按钮的组件：

```
<div x-mojo-element="ToggleButton" id="my-toggle"></div>
```

The x-mojo-element attribute specifies the widget class used to fill out the div when the HTML is added to the page. The id attribute is required to reference the widget from your Javascript and must be unique.

x-mojo-element 属性标识了 HTML 加载到页面时候用来填充 div 的组件的类。Id 属性是组件必须的且是唯一的，用于在脚本中标识组件。

Typically, you would declare the widget within a scene's view file, then direct Mojo to instantiate the widget during the corresponding scene assistant setup method using the scene controller's setupWidget method:

通常要在一个场景的视图文件定义组件，然后在场景助理 setup 方法中用场景控制器的 setupWidget 方法来指导 Mojo 实例化组件。

```
// Setup toggle widget and an observer for when it is changed.  
// this.toggle      attributes for the toggle widget, specifying the 'value'  
//                  property to be set to the toggle's boolean value  
// this.togglemodel  model for toggle; includes 'value' property, and sets  
//                  'disabled' to false meaning the toggle is selectable  
//  
// togglePressed    Event handler for any changes to 'value' property
```



```
this.controller.setupWidget('my-toggle',
this.toggle = { property : 'value' },
this.toggleModel = { value : true, disabled : false });

this.controller.listen('my-toggle', Mojo.Event.propertyChange,
this.togglePressed.bindAsEventListener(this));
```

This code directs the scene controller to setup my-toggle passing a set of attributes, toggle, and a data model, togglemodel, to use when instantiating the widget and to register the togglePressed function for the widget's propertyChange event. The widget will be instantiated whenever this scene is pushed onto the scene stack.

以上代码表述了场景控制器传递了一系列属性和一个数据模型 togglemodel 来实例化和构建 my-toggle 组件。注册了 togglePressed 函数来处理组件的 propertyChange 事件。组件将在每次场景被推出场景堆栈的时候实例化。

To override the default style for this widget, you would select #my-checkbox in your CSS and apply the desired styling (or use .checkbox to override the styling for all checkboxes in your app). For example, to override the default positioning of the toggle button to the right of its label so that it appears to the left of the label:

为了重载组件缺省的风格，需要重新定义#my-checkbox 在 CSS 里面并且提供一个期待的风格（或者用.checkbox 来重载应用程序的所有复选框的风格）例如，为了重载标签右方切换按钮（toggle button）的缺省位置，这样它就显示在标签的左边：

```
#my-toggle { float:left;
}
```

There's a lot more to come so you shouldn't expect to be able to use this to start working with any of these widgets at this point. Chapter 3 and 4 describe each of the widgets and styles in complete detail.

在未来还有很多的内容即将展现，所以你不能期待现在掌握的这些就能随意的使用组件开始你的编程。第三章和第四章详尽的描述了每个组件和风格。

Services 服务

Even limiting yourself to just webOS's System UI, application model and UI widgets, developers would have some unique opportunities for building web applications, particularly with the notification and dashboards. But they'd be missing the access and integration that comes with a native OS platform. The Services functions complete the webOS platform, fulfilling its mission to bridge the web and native app worlds.

即使只想开发 webOS 的系统界面，应用程序模型和用户界面组件，开发者还有少数的机会来创建 web 应用程序，尤其是通知（notification）和指示板（dashboard）。但是这些没有直接访问和集成原生的操作系统平台。服务功能使 webOS 平台更加完善，并成为 web 和原生程序之间完美的桥梁。

Through the Services APIs, you can access hardware features on webOS devices (such as location services, the phone, and the camera) and you can leverage the core application data and services that have always been a key part of a Palm OS device. Almost all of the core applications can be launched from within your application, and there are functions to access data in some core applications.

通过服务的应用程序接口，可以访问 webOS 设备的硬件特性（比如位置信息，电话和照相机），而且可以改变核心程序的数据，并且服务总是 Palm OS 设备的重要部分。几乎所有的核心应用都可以在应用程序（注：指作为服务的程序）中执行，并且有函数来访问某些核心程序的数据。

A service is an on-device server for any resource, data, or configuration that is exposed through the framework for use within an application. The service can be performed by the native OS (in the case of device services), an application, or by a server in the cloud. The model is very powerful as evidenced by the initial set of offered services.

服务是通过框架提供各种资源，数据和配置的单一设备上的服务器程序。服务可以被本地 OS 执行（在作为设备服务的情况下），应用程序或者云计算里面的一个服务器。初期提供的一系列服务已经表明这个模式是非常强大的。

The Services differ from the rest of the framework because they are called through a single controller function, `serviceRequest`. The request passes a JSON object specific to the called service and specifying callbacks for success and failure of the service request.

服务和框架的其他部分不一样,因为它们都是同一个单一的控制函数 `serviceRequest` 来调用。调用请求传递一个 JSON 对象给被调用的服务,并且提供成功或者失败的回调函数给服务请求。

Starting with Chapter 7, you'll find a full description of the general model and handling techniques as well as enumeration of all the services and the specifics for using each one.

从第七章开始将完整的描述通用模型, 处理技术及逐个说明使用的每个服务和具体细节。

Palm webOS Architecture 架构

The Palm webOS is based on the Linux 2.6 kernel, with a combination of open source and Palm components providing user space services, referred to as the Core OS.

Palm webOS 是基于 Linux 2.6内核,组合了开放源码和 Palm 的组件提供用户空间服务(space services), 参考核心操作系统的章节。

You won't have any direct interaction with the Core OS, nor will the end users. Instead your access is through Mojo and the various services. Users interact with the various applications and the UI System Manager, which is responsible for the System UI. Collectively this is known as the Application Environment. Figure 1-11 shows a simplified view of the webOS architecture.

你和最终用户都不会和核心操作系统有任何的直接交互,任何访问都通过 Mojo 和各种服务进行。用户和各种应用程序交互,大家公认用户界面系统管理器是应用程序的环境,也是作为系统的用户界面,图1-11展示了简单的 webOS 架构示意。

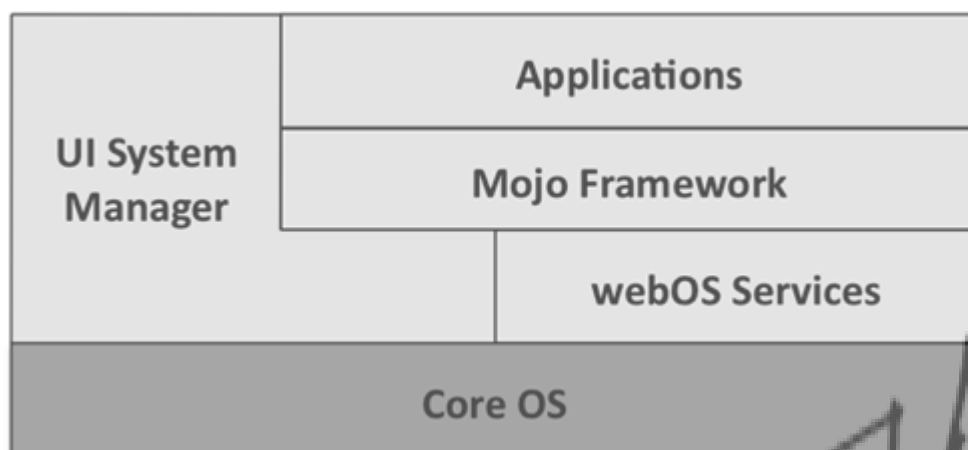


Figure 1-11. Simplified webOS Architecture

This overview is included as background in case you want to have an idea of how webOS works-this information is not needed to build applications so you can skip it if you aren't interested.

作为想对 webOS 如何运转有初步了解的背景资料，这个概述信息不是构建一个应用程序所必须的，所以如果你不感兴趣可以跳过。

Application Environment 应用程序环境

The application runtime environment is managed by the UI system manager, which also presents the System UI manipulated by the user. The framework provides access to the UI Widgets and the Palm Services. Supporting this environment is the Core OS environment, an embedded Linux OS with some custom sub-systems handling telephony, touch and keyboard input, power management, storage and audio routing. All these Core OS capabilities are managed by the Application Environment and exposed to the end user as System UI and to the developer through Mojo APIs.

应用程序运行时环境由界面系统管理器来管理，同时界面系统管理器也作为用户管理系统界面的手段。框架提供用户界面组件和 Palm 服务的访问。在核心操作系统环境通过一个内置的 Linux 操作系统和一些定制的子系统处理电话，触摸屏和键盘输入，电力管理，存储和音频路由，这些来支撑应用程序环境。所有这些核心操作系统功能通过应用程序环境来管理并且作为系统界面展示个最终用户和通过 Mojo API 来展示给开发者。

Taking a deeper look at the webOS Architecture, Figure 1-12 describes the major components

within the Application Environment and the Core OS.

看一下更深入的 webOS 架构,如图1-12 描述了应用程序环境和核心操作系统的主要的部件。

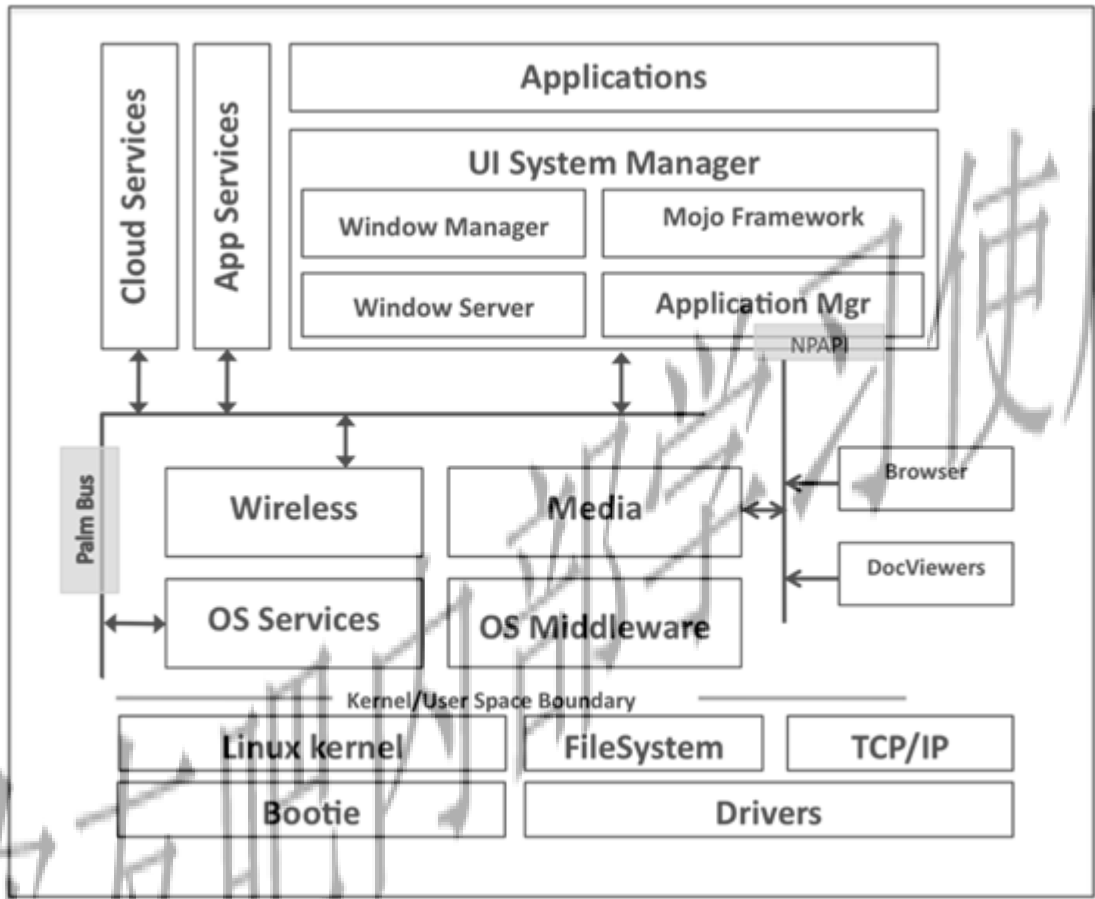


Figure 1-12. webOS System Architecture

The Application Environment refers to the System User Experience and the feature set that is exposed to the application developer, as represented by the Mojo Framework and the Palm Services. The Core OS covers everything else: from the Linux kernel and drivers, up through the OS Services, Middleware, Wireless and Media sub-systems. Let's take a brief look at how this all works together.

应用程序环境参考系统用户体验和 Mojo 框架和 Palm 服务展现给应用程序开发者的特性。核心操作系统覆盖了以下的所有东西：从 Linux 核心和驱动程序，到操作系统服务，中间件，无线和媒体子系统。我们简要看一些所有这些如何协同工作的。

The UI System Manager or UI SysMgr, is responsible for almost everything in the system that is

user visible. The application runtime is provided by the Application Manager, which loads the individual applications, and hosts the built-in framework and some special system apps, the status bar and the Launcher. The Application Manager runs in a single process, schedules and manages each of the running applications, and handles all rendering through interfaces to the Graphics sub-system and on-device storage through interfaces to SQLite.

界面系统管理器（简称 UI SysMgr）几乎处理了系统里所有用户能见的一切。应用程序运行时是由应用程序管理器提供，它加载独立的应用程序并且作为内置框架和其他系统应用的寄主，状态栏和启动器（Launcher）。应用程序管理器单进程运行，它调度和管理每个正在运行的应用程序，通过图形子系统的接口处理所有的渲染，通过 SQLite 的接口来处理设备上的存储。

Applications rely on the framework for their UI features set and for services access. The UI features are built into the framework and handled by the Application Manager directly but the service requests are routed over the Palm Bus to the appropriate service handler.

应用程序交由框架来处理用户界面特性和服务的访问，用户界面特性是在框架内创建并由应用程序管理器直接处理，但是服务请求直接由 Palm 总线路由到相应的服务处理程序。

Core OS 核心操作系统

The core OS is based on a version of the Linux 2.6 kernel with the standard driver architecture managed by udev, with a proprietary boot loader. It supports an ext3 filesystem for the internal (private) file partitions and fat32 for the media file partition, which can be externally mounted via USB for transferring media files to and from the device.

核心操作系统是基于一个版本的 Linux 2.6 内核，udev 管理的标准的驱动架构，一个自带的 bootloader。它用 ext3 的文件系统作为内部（私有）文件分区，fat32 文件系统作为媒体文件分区，这个分区可以通过 USB 外部加载来写入媒体文件或者从设备读取。

The Wireless Comms system at the highest level provides connection management that automatically attaches to WAN and WiFi networks when available, switches connections dynamically, prioritizing WiFi connections when both are available. EVDO or UMTS telephony and WAN data is supported depending upon the particular device model. Palm webOS also supports most standard Bluetooth profiles and provides simple pairing services. The Bluetooth sub-system is tightly integrated with audio routing to dynamically handle audio paths based upon

user preferences and peripheral availability.

无线通讯系统在一个最高的层次来提供连接管理，当可用的时候自动连接 WAN 和 WiFi 网络，动态切换连接，当两者都可用的时候区分 WiFi 连接的优先次序。EVDO 或 UMTS 电话和 WAN 数据的支持主要取决于特定的设备型号。Palm webOS 也支持最标准的蓝牙资料并提供简单的点到点服务。蓝牙子系统与音频路由紧密结合可以根据用户选择和设备可用状态动态的处理语音路径。

The media server is based upon gstreamer and includes support for numerous audio and video codecs, all mainstream image formats, and supports image capture through the built-in camera. Video and audio capture is not supported in the initial webOS products, but is inherently supported by the architecture. Video and audio playback supports both file and stream-based playback.

多媒体服务器基于流，能够支持许多音频和视频编码，所有主流的图形格式，并支持通过内置摄像头捕捉图象。即使架构先天是支持视频和音频捕捉的，但初期的 webOS 产品并不支持这些功能。视频和音频播放支持文件形式和流的方式回放。

Software Developer Kit (SDK) 软件开发包

Of course the best way to get started writing webOS applications is to continue reading this book, but you should also go to Palm's developer site, <http://developer.palm.com> and register as a Palm developer and download the Palm Software Developer Kit (SDK). The SDK includes the development tools, sample code, the Mojo Framework, along with access to the Palm Developer Wiki, where developers will find formal and informal training materials, tutorial and reference documentation. Palm also provides registered developers with direct technical support by email or through interaction in a hosted developer forum.

当然，开始编写 webOS 应用程序的最好途径是继续读本书，但你也应该去 Palm 的开发者网站 <http://developer.palm.com>，并注册和下载 SDK。SDK 包括了开发工具，例子源代码，Mojo 框架，通过访问开发者 wiki，开发者将获得正式的、非正式的练习资料和参考文档，Palm 也为注册开发者提供 email 支持，或在主要开发者论坛上面提供交互的支持。

Development Tools 开发工具

The Palm Developer Tools (PDT) are installed from the SDK and include targets for Linux, Windows (XP/Vista) and Mac OS X. The tools enable you to create a new Palm project using sample code and framework defaults, search reference documentation, debug your app in the weOS emulator or an attached Palm device, and publish an application. Chapter 2 includes more details about the tools in Palm's SDK and third-party tools, but you'll find a brief summary in Table 1-1 below.

Palm 开发者工具从 SDK 安装，包括 Linux、Windows (XP/Vista) 和 Mac OS X。这个工具允许你用样例代码和框架缺省设置创建新的 Palm 工程，查找参考文档，在 webOS 模拟器或者 Palm 设备上调试你的程序，发布应用程序。第二章将详细的讨论 SDK 里面的工具和第三方工具，不过你可用从下面的表中看到简单的概括。

Table 1-1. Palm Developer Tools

Tools	Major Features
SDK Bundle Installer	Installs all webOS tools & SDK for 3 rd party editors
Emulator	Desktop Emulator and Device Manager
Command-Line Tools	Create New Project
	Install & Launch in Desktop Emulator or Device
	Open Inspector/Debugger Window
	Package & Sign App

工具	主要特征
SDK 安装包	为第三方编辑器安装所有的 webOS 工具和 SDK
仿真器	桌面仿真器和设备管理
命令行工具	建立新的工程
	安装和在桌面仿真器或者设备里执行
	打开检查/调试窗口
	打包和签名应用程序

The tools can be installed and accessed as command-line tools on every platform and include some bundles for integration into popular HTML editors and as a plug-in to Eclipse and Aptana Studio, a popular Javascript/HTML/CSS editor for Eclipse. Refer to Palm's Developer portal for

the most current list of supported editors and tool bundles.

开发工具在所有的平台上都可作为命令行的方式进行安装和使用,包括一些集成到流行的 HTML 编辑器和 Eclipse 插件和 Aptana 工作室(一个流行的 Javascript/HTML/CSS Eclipse 编辑器)。更多更准确的编辑器和开发工具清单参考 palm 的开发者门户。

Mojo Framework and Sample Code 框架和样例代码

The SDK installation includes a copy of the Framework and sample code to help you design and implement your application. Unlike most JavaScript frameworks, you won't need to include the Mojo framework with your application code since Palm includes the framework in every webOS device. The framework code included in the SDK is for reference purposes to help you with debugging your applications.

SDK 的安装包括一个框架的拷贝和用于帮助你设计和实现自己应用程序的样例代码。不像其他多数的 JavaScript 框架,你不需要在应用程序代码中包含 Mojo 框架, Palm 已经包含框架在所有的 webOS 设备里面。SDK 里面包含的框架代码用来参考以帮助你调试应用程序。

The sample code is also for reference. There are samples for most of the significant framework functions, including application lifecycle functions, UI widgets and each of the services. Simple applications are included to give you some starter applications to review and leverage as you choose.

样例代码也是用于参考,代码有很多框架的重要函数功能的例子,包括程序生命周期函数,用户界面组件和每个服务。简单的应用程序包括一些初学者程序来回顾

Developer Portal 开发者门户

Your main entry point is <http://developer.palm.com/>, which is where Palm hosts the Developer Portal. The portal provides access to everything that you might need to build webOS applications, including access to the SDK, all development tools, and documentation and training materials.

主要的入口是 <http://developer.palm.com/>，Palm 作为开发者的门户网站。门户提供了访问生成 webOS 应用程序所需的所有东西，包括访问 SDK，所有开发工具和文档，培训教材。

The Developer Portal provides your application signing services and access to the Application Catalog. This is an application store that is published and promoted with every webOS device through a built-in App Catalog application. Applications need to be signed for installation on a webOS device, and through the portal you can access the signing tools and related support.

开发者门户提供应用程序签名服务和应用程序编目访问。这是一个应用程序商店，通过内建的应用程序编目程序发布和推销到每个 webOS 设备。应用程序需要签名才能安装到 webOS 设备，通过门户可用访问签名工具和相关支持。

Summary 总结

In this introductory chapter, you were introduced to webOS, Palm's next generation operating system. The following chapters will cover each of these topics in far more detail but this chapter should have helped you understand the webOS architecture and application model along with the basic services available in the SDK.

在这个介绍性的章节，介绍了 webOS，Palm 的新一代操作系统。接下来的章节将尽可能详细的覆盖这些主题，而这一章可以帮助理解 webOS 的架构应用程序模型除了 SDK 里面可用的基本服务。

You'll find that it's pretty easy to get started writing webOS applications. After all, you're simply building web applications, using conventional web languages and tools. You can port a very simple Ajax application by creating an appinfo.json file for your application at the same level as your application's index.html file. With as little as that, your app can be published and available for download to any webOS device.

你将发现开始写 webOS 程序是相当的容易，之后，你将很简单的用传统的 web 语言和工具建立 web 应用程序。通过建立一个与你程序的 index.html 文件同级的 appinfo.json 文件，你就可以发布一个非常简单的 Ajax 应用程序。

From there you can invest more deeply by building in the Mojo UI widgets to take advantage of the fluid physics engine, gesture navigation, beautiful visual features, text editing, and the powerful notification system. You can move beyond simple foreground applications that rely on active user interaction, and adapt your application to run in the background or even be headless. Or consider an application that can open new windows for each new activity, enabling the user to multi-task within a single application. There's a whole new generation of applications possible on the webOS platform, just waiting to be built.

从那之后，你可以进一步用 Mojo 用户界面组件来构建程序，以获得液体物理引擎（fluid physics engine），手势导航，漂亮的可视化特性，文本编辑和功能强大的提醒系统。你可以跳过简单的与用户交互的前台应用程序，改写你的程序成为运行在后台或无标题的。或者试图做一个程序可以为每个活动打开新窗口，允许用户在一个程序里面进行多任务。这些是在 webOS 平台上的整个新一代的应用程序，等着我们去构建。