

第14章 无线应用协议

传统意义上，当提到 Web 时，一般指的是桌面计算机上的浏览工具。但这个概念目前正在逐渐发生改变，现在的一些消费产品如移动电话、个人数字助理（或者是类似 Palm Pilot 和 Psion Organizer 的 PDA）等也提供了 Web 浏览功能。但这些产品在低带宽、小屏幕的基础上运行，因此我们不得不重新考虑网络服务的生成和传送方式。虽然这个市场目前还没有成熟，但现在已经产生了一种与传统浏览或冲浪不同的服务传输方法及相应的标准。

1998 年度移动电话的销售量超过了桌面电脑，许多人预测这个趋势将一直持续到 21 世纪。由于众多的产品都提供了桌面电脑的许多功能，Internet 将会日趋多样化，那些希望脱颖而出的网络服务必须具有能够适应广泛的客户群体的能力。

在这一章里我们将讨论这些新技术给网络开发者带来的问题，以及当前出现的称为无线应用协议的一组标准，这组标准是为具有网络功能的移动电话和 PDA 设计的。无线应用协议（Wireless Application Protocol, WAP）的网站是 <http://www.wapforum.org/>。我们也将讨论一些 WAP 相关技术，尤其是在网络设备不仅仅局限于桌面系统的“未来平台”时，XML 如何为内容提供的一个坚实基础。

为了介绍这些今后的应用趋势，我们也会讨论 XML 的一个应用——无线标记语言（Wireless Markup Language, WML）。WML 是 WAP 中用来标记传输给移动电话、PDA 等设备的数据的语言部分。

本章中我们着重讲述：

- 新产品给网络开发带来的挑战。
- WAP 如何为这些新产品提供网络内容服务。
- 简介无线标记语言。
- 在移动电话上一个搜索 Wrox 图书目录的用户界面。
- 如何使用 XML 实现为不同的平台提供页面而无须单独为每个平台开发不同的站点。
- 简介无线标记语言脚本——WAP 的脚本语言部分。

如前所述，我们认为这些新的消费产品会成为 Web 的未来发展趋势，下面探讨一下在无线领域里我们将会面临的挑战。

14.1 新客户介绍

同桌面计算机相比，新一代具有网络功能的产品无论是硬件、软件还是网络功能都很有限。为了给所有的 Internet 客户提供合适的服务——从移动电话到桌面计算机——网络服务提供商不得不根据客户的功能特点调整他们的服务。

目前，大多数网络服务商都对所有的桌面系统客户提供大致相同的服务。一些网络服务商对不同的浏览器提供同一网络站点的不同版本，比如一个版本是为较新版的 Internet Explorer 设

计的，一个是为 Netscape Navigator 4 设计的，另一个则是为老版本的浏览器设计的。对那些曾经为不同版本的桌面浏览器提供具有同一功能的网络页面而头疼的网络服务提供商来说，为这些新一代产品提供服务无疑是一场灾难。但我们在下面的“从底层开始建立站点”这一部分将会看到，实际情况并非如此。

随着 Internet 日益变得丰富多样，网络服务商必须在信息提供上越来越智能化，能够根据不同产品的功能提供客户想要的信息服务。虽然移动电话和 PDA 同桌面系统并行发展，它们的存储容量及处理能力都会日益提高，但它们同桌面系统的能力差别还是很大。这就意味着调整网络服务使其更适应广泛的设备需求不会是一个暂时性的任务，这是今后 Web 发展的一个事实。

新产品的普遍特性包括：

- 显示屏幕小，分辨率低。
- 存储容量和处理能力不足以支持软件运行和应用系统存储。
- 同有线网络相比，网络带宽较窄。

由于显示能力和网络带宽也有差异，我们不仅要为特定的产品提供不同的表示语言，网络服务的结构和内容也有可能不尽相同。例如，为了适应移动电话的需求，你可能会将“请输入你的姓名”这句话改为“输入姓名”。

另外，这些又产生了其他一些有趣的领域；例如，不同显示意味着我们需要鉴别新的样式和不同的 UI 原型。而且，支持移动设备的各种技术也使网络应用产生了新的方向——从在移动过程中确定方位到推技术和语音技术的新应用。

消费产品市场正以惊人的速度增长，因此，“过时”产品的数量将极其庞大。为了适应众多的产品需求，网络内容必须以一个独立的格式存储，这种格式能不受表示语言和产品功能飞速发展的影响。

那么，如何解决建立一个站点的众多不同版本这个问题呢？答案是要采用一种不同的方式建立网络站点。

从底层开始建立站点

一直到最近，绝大多数网络页面都是用 HTML 写的，主要用于在桌面浏览器上显示。HTML 是能够有效地描述桌面网络浏览器显示页面的一种语言。HTML 着重于描述，而不是内容。这一点正是在网络开发中应用 XML 的主要优势之一。

如果将网络站点看作是内容，用 XML 标记数据部分，就可以将同一数据转换为不同的格式。也就是说可以通过改变显示的内容以及显示内容的方式来有效地适应不同客户的需求。就像在第 9 章——操作 XML 中看到的，我们可以使用 XSL 将用 XML 标记的站点内容转换成不同的 HTML 版本，也可以用 XSL 将内容转换成其他表示语言。对于希望为传统网络客户以外的客户群提供服务的网络服务商来说这是一个理想的解决方案。要想为不同的产品提供相应的服务，只要改变 XML 文档就可以了（我们在案例 3 中将探讨另一种转换技术，这种技术可能更适应大的文件）。

使用 XML 语言写内容，可以使用模板来针对不同的产品提供相应的服务。我们正在从底层开始建立站点：首先建立内容（站点的核心），然后根据不同的目的改变内容的格式。支持一种新产品或一种新版语言就意味着生成新的模板，而不是去复制以某种表示语言的某一版本形式存贮的大量文档。这意味着我们只需为每一个目标提供合适的内容——这些内容仍然支持桌面

系统客户，但同时也考虑了新的手持产品的限制和发展趋势，当然数据将会以不同的形式出现。

当查询一个餐馆目录时，不同的移动设备很难在一个屏幕上显示同样的数据，因此用户可能会采取不同的方式找到所需的数据（参见图 14-1）。提供的服务也可能会扩展到寻找一条到达指定餐馆的最佳路径——根据移动用户的当前位置。但最根本的需求仍然是提供任何一个用户所需要的餐馆的详细信息。

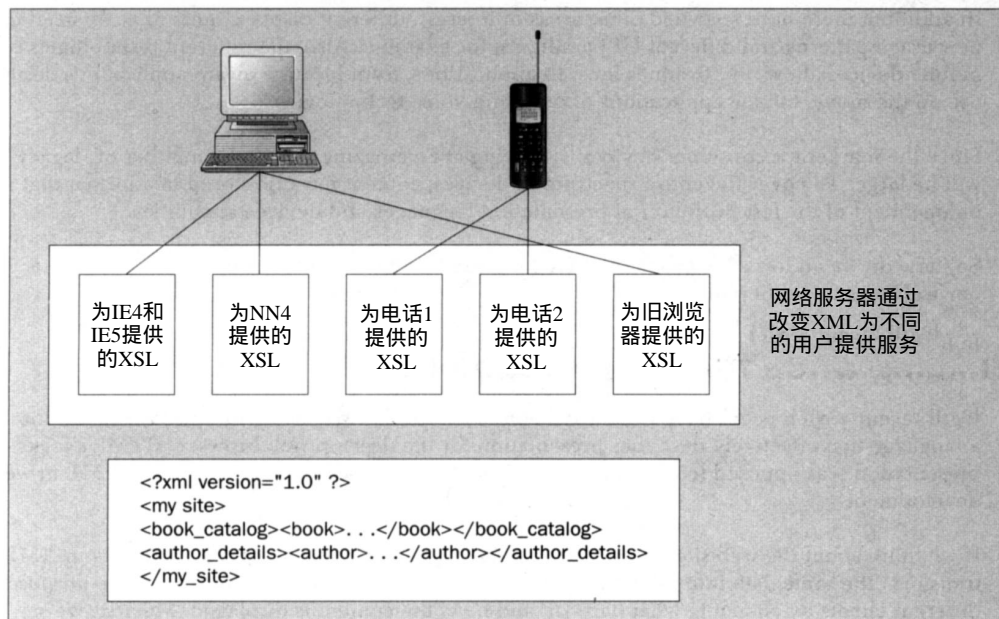


图 14-1

(1) 向电话传送数据

这一章我们将说明如何借助 XML 实现将 Web 扩展到无线网络和产品上。HTML 4.0 提供的一套丰富特性并不适合这些产品，因而使得标准表示语言不适用于移动电话和 PDA。目前，适用于手持客户的表示语言有：

- 各种 HTML 专有子集（其中一些带有专有扩展）。
- 手持产品标记语言（Handheld Devices Markup Language, HDML）。
- 无线标记语言（WML）。

在上述三种语言中，移动电话厂家最支持 WML。WML 是由无线应用协议论坛（WAP Forum）定义的，这个论坛成立于 1997 年，由 Ericsson、Nokia、Motorola 和 Phone.com 共同建立，目前它已经拥有 250 多个成员，是将 Web 扩展到无线网络的最成功的倡导者。WML 为内容开发商提供了适应于全世界移动电话和各种不同的网络协议的一个统一接口。

HDML 是由 Phone.com 定义的一种专有语言，只有使用 Phone.com 浏览器的移动电话支持它。HTML 的一个子集“压缩 HTML（Compact HTML）”早在 1998 年就提交给 W3C。在日本压缩 HTML 成功应用于 NTT DoCoMo（世界上最大的网络接入者之一）提供的“i-模式”服务中。但在这一章中，我们主要讲 WML，因为所有的电话生产者都支持它，并且它也是 XML 的一种应用。

毕竟，压缩 HTML 并没有任何 HTML 中没有的东西。但 WML 却是一个有趣的而且具有许多 HTML 没有的新特性的一个小工具箱。

虽然 XML 不能使网络速度更快或使移动电话的显示屏幕更大，但如果用 XML 代替 HTML 编写服务程序，同一种服务可以根据客户的不同接受能力（在向服务器发出的请求里报告）产生几个不同版本。要想成功地完成上述功能，不仅要了解 XML，而且还要了解移动电话作为 Internet 的一个客户所具有的特点。

这一章讲述内容开发商所需要了解的无线网络特性和表示语言——WML。正如 HTML 对桌面计算机的作用一样，XML 的这一应用——WML 是移动电话显示的一个接口。

(2) WAP 的支持技术

WAP 不只包括一个规范，而是包括一组规范，并且这组规范还会随着技术成熟而进一步扩展。下面的规范组成了 WAP 1.1 的一部分。不要被这么长的清单吓倒，通常内容开发商并不需要了解所有这些内容，因为 WAP 对大多数开发商屏蔽掉了底层的网络和服务提供的细节。但正是这组协议的存在才使软件开发者能够提供相应的软件。这里我们打算研究全部的内容，而是主要讨论那些用来为具有 Web 功能的产品提供服务的协议。

- 无线应用环境（Wireless Application Environment, WAE）规范——全部的应用环境。
- WML 规范——将 XML 转换成传输到移动电话的格式的一种标记语言。
- 二进制 XML 内容格式规范——一种字典式压缩方法，可以对 XML 文档如 WML 文档进行编码。
- WMLScript 语言规范——一种类似 JavaScript 的语言，对内存 CPU 的使用率最小。
- WMLScript 标准库规范——所有客户都支持的标准脚本库。
- WAP 缓存模型规范——浏览器的缓存及它与浏览器浏览记录的关系。
- 无线会话协议（Wireless Session Protocol, WSP）规范——Internet HTTP 协议的二进制版本。
- 无线交换协议（Wireless Transaction Protocol, WTP）规范——WSP 的辅助协议，用来管理单独的请求-应答交换。
- 无线数据报协议（Wireless Datagram Protocol, WDP）规范——各种无线传输服务的公用接口。
- GSM USSD 的无线应用协议的规范——无线数据报协议对 USSD 的映射，这是在 GSM 网络中使用的一种传输服务。
- 无线控制信息协议（Wireless Control Message Protocol, WCMP）规范——用于在不支持 Internet 控制信息协议（Internet Control Message Protocol, ICMP）的网址上报告错误。
- 无线传输层安全（Wireless Transport Layer Security, WTLS）规范——用于认证和加密。
- 无线电话应用（Wireless Telephony Application, WTA）规范——使进来和出去的电话通过无线标记语言和无线标记语言脚本来控制，允许信任伙伴如网络接入者采用语言和 Internet 混合服务——其用途包括自动呼叫一个在黄页查询中找到的号码，以及语音邮件系统的可视接口。
- 无线电话接口（Wireless Telephony Application Interface, WTAI）规范——WMLScript 库到

电话服务如电话簿和呼叫日志。

前面我们已经讨论过为了新环境的应用需求开发者必需了解的概念，下面我们看看如何实现无线应用协议。

14.2 了解无线环境

在过去的10年左右时间里，无线应用领域的急剧发展令半导体行业也取得了显著的进展。因此，电话和PDA越来越小，越来越轻。这个趋势还会以一定的速度持续，直到显示屏幕和键盘不能变得更小。

此外，这些移动产品的带宽同我们连接到Internet上的有线网络（采用线缆传送数据的网络）相比很窄。一个数据呼叫的带宽可低至9 600bps。延迟时间——一个数据包从一处传递到另一处的时间——将会使用户调用应用程序的过程不太令人满意，尤其是当开发者没有将应用程序要求的网络回合数目降至最小时这种情况将更为严重。

对网络开发者来说，绝大多数用户使用存储容量和处理能力都很有限的小电话。这些小产品的显示屏幕和键盘也很小。因此，成功的应用应该将这些特性考虑进去。为了使用户快速完成任务以及使点击次数最少，以下几个参数必须最小化：用户手敲入的文字数量，显示的信息量，连接服务器的次数，以及图形的尺寸。与此同时，PDA市场有可能出现相对更高配置的产品，如处理器和存储器。因此，不同产品之间的差别将会变大。另一方面，在2000年初购买WAP电话的用户会希望其电话中内置的WML浏览器至少能提供3年的服务，也就是到2003年。然而到了2003年，人们预测移动电话和网络都会比现在的功能强大得多。这个“遗留问题”在传统Web上也存在——1995年的HTML浏览器不能显示现在的许多网络站点，但因为浏览器可以从网络上免费下载并安装，这个问题并不很突出。

网络开发者将会面临的问题是——屏幕尺寸、提供的带宽、分辨率、处理能力、存储能力、键盘分布、语言、版本等都处于不断变化之中。

14.2.1 服务

那么，未来我们希望在移动客户上见到什么样的服务呢？答案无疑是没有界限的——WAP可以提供在移动过程中可能需要的任何信息服务，或者任何在现有的电话服务上使用的功能。一些移动电话和PDA上较普遍的Web服务包括：

- 黄页。
- 股票报价。
- 预定航班。
- 移动银行系统。
- 群件系统。
- 地址簿。
- 预定晚餐。

上述这些服务不同于我们所了解的浏览或“冲浪”，因为通常用户知道他们正在寻找的信息类型。

一个成功的无线应用程序的关键在于能对所需信息进行快速而简便的查询。

对开发者来说,这说明好的服务必须是简单的,而且应该尽可能少地使用网络资源。

大多数 WAP 浏览器提供商有 WAP 的在线演示,相关信息可以参阅本章结尾。

下面我们将生成一个应用程序,它可以使用户从 Wrox 图书分类目录中查找作者姓名。图 14-2 是我们生成的应用程序的屏幕显示。

我们看到,这里的信息极其简单,仅保留了基本内容。最根本的一点是用户应该得到他们想要的信息——多媒体显示可能会满足具有高带宽网络连接的桌面系统用户的需求,但通过窄带宽(比拨号网络更糟糕)连接的移动用户不会喜欢这些。此外,无线连接通常比有线服务更昂贵。因此当为移动电话提供服务时,应该更多地考虑带宽需求,“节省带宽”的应用程序不仅运行更快,而且会为用户节约费用。

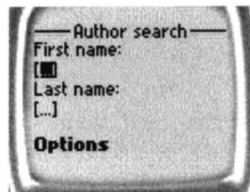


图 14-2

这里我们强调了无线技术目前的局限性,同 Internet 其他领域的发展一样,毫无疑问服务性能会呈上升趋势,而费用会逐渐下降。

前面已经提到,我们无须了解 WAP 规范的各个方面,因为 WAP 对内容开发商屏蔽了不同网络技术的细节。但是,无线网络的特性相对于其应用程序并不透明。下面我们将详细地讨论无线网络的特性。

14.2.2 无线网络

为了了解幕后 WAP 是如何工作的,我们有必要看看由传统有线网络构成的 Internet。通过分析有线到无线技术转移过程中遇到的问题,将会看到 WAP 对这些问题的解决方案。

数据怎样通过 Internet

Internet 产生时,人们定义了一系列同时运行于网络上的协议。这些协议叫做 Internet 协议族。协议族中最常见的两个协议是 TCP (Transmission Control Protocol, 传输控制协议) 和 IP (Internet Protocol, 网际协议),其中 TCP 提供基于连接的可靠字节流式服务,对源和目标之间的数据传输进行正确校验,IP 提供数据包路由服务。

IP 很少一次传送全部数据,而是将数据信息打成更小的称为“数据报”的信息包进行传递。这样就无须总是和 Internet 连接,而是当有数据包传送时才连接。如果一个数据包没有传送到目的地,TCP 就会知道并对其重新传送。

在无线网络中可以像在有线网络中一样使用调制解调器,调制解调器建立了一个连续的通道,IP 数据报可以通过这个通道进行传输。通常在连接过程中会有很长的时间内没有数据传输(例如当用户阅读一个已经下载的文档时),但即使是在没有数据传输时调制解调器也要保持连接。在一个费用昂贵的无线网络中这种做法就很不经济了,因此那些只有在需要时才传输数据的网络正变得越来越普遍。最通用的无线数据包网络是蜂窝数字分组数据系统 (Cellular Digital Packet Data, CDPD) 和通用分组无线服务系统 (General Packet Radio Service, GPRS)。美国和新西兰广泛采用了 CDPD 系统,而 GPRS 则发展成为全球通系统 (Global System for Mobile

Communication, GSM)标准的扩展,并在欧洲、亚洲和美国部分地区成为主要的移动电话数字系统。

(1) 数据怎样在空气中传输

这些年来,无线领域采用了几种不相容的技术。首先,无线领域中采用了模拟和数字系统。美国最著名的模拟系统是 AMPS——标准模拟电话系统。但目前新生的数字系统正在迅速地代替模拟系统。模拟系统中每个用户拥有一个特定的无线频率。网络服务商可使用的频率是有限的,因此这种资源的使用很昂贵。数字系统的一个好处就是多个用户可以同时共享一个频率,这样电话服务商就可以更好地利用网络,从而为语音和数据提供更多的带宽。多个用户共享一个频率的方法有以下两种:

- 时分多址复用 (Time Division Multiple Access, TDMA)
- 码分多址复用 (Code Division Multiple Access, CDMA)

GSM是TDMA系统,但在没有GSM的地方,CDMA系统正日益普遍起来。我们无须知道这些系统工作的技术细节,因而WAP的一个最重要的目标就是使不同网络的技术细节对开发者透明。应用开发者基本上只需要知道一个给定的支持网络和电话的WAP就可以了。关于底层网络如何和电话之间传递数据的规定已经被生产者在WAP标准中规定好了。当用户通过浏览器访问Web时,电话会自动利用其中一项无线技术连接到网络上。

我们稍后可以看到,WAP采用了一种和Web相似的应用环境,使得为无线产品开发应用程序和为桌面系统开发连接Internet的应用程序一样容易。在WAP以前,要根据不同的网络采用不同的技术提供服务,甚至在一些网络中无法添加声音以外的其他服务。例如在GSM中,许多服务是用SIM应用工具箱生成的,这些用SIM应用工具箱生成的服务安装在每个GSM电话带的SIM卡上。这种方法对一些服务很适合。但这些服务只能用于GSM网络中,而且开发和共享都不方便,因而无法提供像WAP浏览器那样的复杂用户界面。

(2) 无线网络上的其他服务

无线网络还提供一些与标准Internet协议非常不同的数据服务。比如,GSM短信息服务(GSM Short Message Service, GSM SMS)和寻呼网络。SMS服务最初是用于在GSM系统中提供分页服务,但由于一些具有创新精神的公司的努力,它现在已经成为一项通用的数据包服务。一个SMS数据包最多有140比特,其延迟时间可以为2秒钟到几个小时。因为SMS不是设计成一个通用的数据包网络,而是一个同分页网络相似的短信息服务,所以其延迟时间很长。

(3) 在无线网络上使用TCP/IP出现的问题

当使用调制解调器时,TCP/IP——标准Internet协议族——可以用于无线网络中,但可以使用的带宽却比有线网络低很多(如同我们前面提到的)。由于移动和无线环境无法预测经常出现的问题——如线路连接质量不好,甚至会出现暂时的连接中断现象。有时当用户通过一条隧道时,他可能超出了无线覆盖范围。即使这种问题不会影响语音通话的质量,它也会降低TCP协议的性能,因为TCP协议是为具有稳定连接的有线网络设计的。当连接中断时,TCP假定网络发生阻塞,因而会显著降低传输速率,当连接恢复时,TCP需要一段时间才能回到最初的高传输速率。所以,在无线网络中,TCP的表现并没有在有线网络中好。在过去的2年里许多大型研究实验室都致力于优化无线网络上的TCP。尽管已经取得了一些成功,但这些进展无论是在传输速度上还

是可靠性上都不十分显著，并且标准 TCP/IP 软件被广泛使用，因此目前还没有一个“无线 TCP”能够取代标准 TCP 的位置。

新的网络技术为移动用户带来了更宽的带宽。但这个带宽究竟有多宽，费用如何等都还没有确定。究竟下一代的网络技术是为每个人提供一个合理价位的带宽，还是仅仅为少数有支付能力的移动专家们提供昂贵的带宽呢？我们还要拭目以待。在无线网络上传输数据需要一定范围的无线频率。传输数据越多，所需频率就越多，但无线频率是一个有限的资源，这个问题同有线网络不一样，因为在有线网络中你只需要增加线缆就可以了。

下面总结一下开发无线应用程序的障碍：

- 众多不同的网络技术使得服务提供的方式各不相同，并且互相不兼容。
- 固有的长延迟时间和有限的无线频率使得带宽成为一个问题。
- 由于数据和无线频率的双重作用，对带宽要求高的网络服务变得很昂贵，并且很难降低。
- 网络服务质量依赖于网络提供商和无线电技术。

14.2.3 WAP 如何解决无线网络应用遇到的问题

WAP 论坛的目的是将其会员公司各自拥有的移动网络信息技术统一起来。如前所述，WAP 论坛成立于 1997 年，由 Ericsson、Nokia、Motorola 和 Phone.com 共同发起。到 2000 年 1 月 WAP 论坛已经拥有 250 多个成员。

在 WAP 论坛成立以前，网络技术在无线网络方面没有太大的进展。前面已经提到过，传统 Web 上的许多技术很难适用于移动电话。但世界上移动电话的数量正在逐渐增长，而 Web 上的信息也急剧膨胀。很明显下一步的任务就是在这两个市场结合起来。那么，如何使移动设备的用户克服接受技术的限制去访问网络信息资源呢？

网络协议如 HTTP 和语言如 XML 设计为具有可读性。这一特性使得开发和测试过程简单化，同时也是这些协议的成功原因。但这一特性也使得它们非常浪费带宽，文本处理协议要求大量的字符串比较和字符串管理。例如，如果一个网络应用程序是 10KB，则 HTTP 的头部信息会有 100 或 200 比特，这个数字在有线的 Internet 上算不上什么。但是在性能大大降低的无线应用环境中，一个相应的应用程序一共不过 500 比特，那么一个 200 比特的头信息就会有很大影响。

为了解决这些问题——低带宽、延迟时间长等，也为了使不同无线网络的特性对网络服务器透明，WAP 定义了移动电话和网络站点上网络服务器之间的一个网关。在 WAP 网关和网络服务器之间采用了标准网络协议（如 XML、HTTP 和 TCP/IP）。在移动电话和网关之间则使用 WAP 论坛规定的协议。这些协议是网络协议的二进制版本，通过压缩数据量使移动电话能够访问网络。如图 14-3 所示，网关负责不同协议的转换。

WAP 中除了 WAP 网关和移动电话之间进行通信服务的二进制网络协议外，还有更多的其他协议。我们讨论完这些协议后，将会很快重新对在 WAP 模型中如何提供数据进行讨论。

网关可能还具有其他一些功能，比如对不同的字符集进行互换编码。一个字符集定义了数字和字母之间的映射。例如 84 可能代表字母“T”。不同的语言和国家可能会使用不同的字符集。

WAP 使用的另一版本的 HTTP 是无线会话协议（Wireless Session Protocol, WSP），无线会话

协议使用二进制方式传输信息，而不是使用文本格式。同 HTTP不同的是，它并不要求一个面向连接的协议（TCP）提供底层服务，却可以在任何一种数据报服务如 SMS和UDP（Internet上的数据报服务）上进行。由于数据报服务是不可靠的，无线会话协议采用面向传输的 WTP协议管理每一个请求-应答的传送，必要时重新发送信息。二进制协议 WSP和文本协议 HTTP的区别在于：如果HTTP的头信息是一个包括25个ASC 字符的字符串：

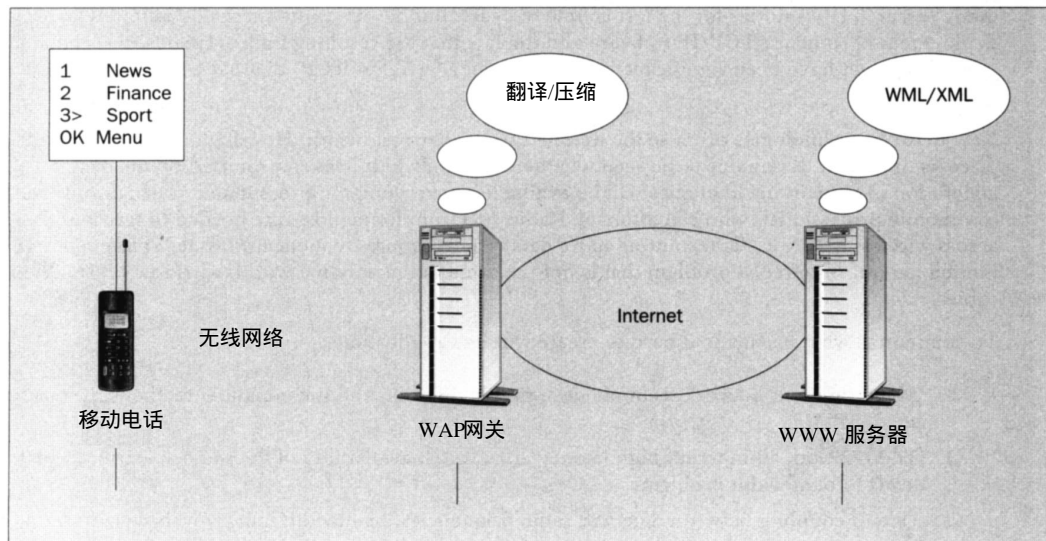


图 14-3

```
Accept-Language: en;q=0.7
```

则相应的WSP的头信息会是4个字节：

```
0x83 0x02 0x99 0x47
```

在WSP标准中定义了字符到二进制的映射。这样，在 WAP网关和发出请求的用户之间传输的数据就会少得多了。WAP网关主要用于这两种格式互相转换。作为一个服务器应用程序的开发者，你不必考虑格式转换问题，WAP网关已经将这个过程全部完成了。我们所编写的 WML也同样要进行字符到二进制的转换。例如，下面的 WML文档：

程序清单 14-1

```

<wml>
<card id="abc" ordered="true">
  <p>
    <do type="accept">
      <go href="http://xyz.org/s"/>
    </do>
    X: $(X) <br/>
    Y: $(&#x59;) <br/>
    Enter name: <input type="text" name="N"/>
  </p>
</card>
</wml>
  
```

经过WAP网关变成下面的文档：

程序清单 14-2

```
02 08 6A 04 'X' 00 'Y' 00 7F E7 55 03 'a' 'b' 'c' 00
33 01 60 E8 38 01 AB 4B 03 'x' 'y' 'z' 00 88 03
's' 00 01 01 03 ' ' 'X' ':' ' ' ' ' 00 82 00 26 03 ' ' 'Y'
': ' ' ' 00 82 02 26 03 ' ' 'E' 'n' 't' 'e' 'r' ' ' 'n'
'a' 'm' 'e' ':' ' ' ' ' 00 AF 48 21 03 'N' 00 01 01 01 01
```

其中所有的XML标记和属性都用二进制数值代替。这样，移动电话并没有真正接收到 XML，而是接收到一系列经WAP网关转换的XML文档的二进制值。

除了将请求和应答信息中的 HTTP头信息转换成二进制值以外，WAP网关还有其他的优点。许多HTTP头信息在各条信息中都是相同的，网关就可以储存头信息，这样浏览器就无须一遍遍发送这些信息，因而节约了带宽。网关还支持“推”技术，也就是说无须移动电话请求，网关可以主动向它发送数据。这项技术同现在的寻呼机服务很相似——新闻和体育赛事结果可以直接传送（推）到用户那里。

前面我们讨论了如何把数据传递到移动产品上，并且了解了为这些移动用户提供服务涉及到的一些问题，下面我们要讨论一下将信息传递给用户的标记语言。

14.3 介绍WML

WML是XML的一个应用，它包括 XHTML的部分元素，支持表达式和变量、用户输入认证以及一个通用的事件模型。

WML同HTML 4.0相比是一个小型语言，但对于我们所需开发的应用来说已经足够了。同桌面计算机系统相比，我们面向的对象屏幕更小，分辨率更低，因此 HTML的许多特性对我们来说没有必要。但WML也添加了一些HTML中没有的特性，如文档和状态管理的参数变量，这些特性可以使不同类型网络资源的使用更加灵活。

因为WML主要是为前面提到的小型产品所设计的，因此它的规则制定要遵循如下几个规则：

- 显示屏幕小，用户功能有限。
- 网络连接带宽窄。
- 存储容量小，计算能力有限。

这几个原则恰好反映了我们在本章前面所提到的问题。

这一部分是WML特性的一个概述，尤其着重于HTML中没有的一些特性。虽然WAP的第一版很稳定，但为了同加速发展的市场保持同步，WAP很快就会推出第二版。这一部分里描述的WML是1.1版。还需要提到的一点是，如果你用XML写的内容建立站点，如同我们在“从底层开始建立站点”那部分所描述的那样，你有可能不用WML写网页，而是用转换语言如XSLT将现有的数据转换成WML格式。但无论怎样，你也需要了解WML，这样才能将XML文档转换成WML文档。

14.3.1 怎样将第一份文档传送到电话上

同Web上的其他浏览器一样，可以从 URL上请求访问WML文档。也同所有的HTML浏览器

一样，WML浏览器也可以通过某种途径输入 URL，其具体的实现机制则取决于浏览器。因为通过电话键盘输入长串的 URL 十分不方便，因此浏览器厂商为用户提供了配置浏览器的其他方法。但无论如何，电话上的网络浏览器也并不意味着你可以“打电话”访问网站并得到传输回的内容。毕竟它还是 Web，它的工作方式也同 Web 相似。

14.3.2 WML文档的结构

HTML 将文档分为 <head> 和 <body>，整个文档作为一个网络页面同时传送。同 HTML 不同，WML 将其信息组织结构比喻为卡片组 / 卡片 (deck/card)。卡片组表示组成某一特定 URL 上的资源的卡片集合 (卡片组的作用同 HTML 页面差不多)。WML 元素在一个文档中传输一个或多个卡片 (卡片同 HTML 中的帧不同的是，卡片通常不同时显示)。同单独下载许多小文档相比，在卡片组中包括一组卡片可以节省网络时间。下面是一个 WML 文档整体结构的范例：

程序清单 14-3

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>

    <head>
    ...
    </head>

    <template>
    ...
    </template>

    <card>
        <p>
        ...
        </p>
    </card>

    <card>
        <p>
        ...
        </p>
    </card>
</wml>
```

为了更仔细地研究一下，我们从上面这个文档的头部开始看。WAP 网关在文档传给移动电话前要对其进行确认，所以文档到达网关时必须提交文档头：

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">
```

为了节约篇幅，下面的例子中都不再有文档头，但它实际上是存在的。由于网关对文档进行确认，劣质文档或无效文档会被拒绝传送给客户。因此，在将文档传给无线网络之前对其进行校验无疑是正确的。

文档头后紧跟根元素——<wml>，后面的 <head> 和 <template> 元素是可选项，因此最小的

WML是一个只有一个卡片的文档。

这里提到的<head>元素同HTML文件中的<head>元素功能相同，都包括了文档的各种信息。HTML文档中的<meta>元素包含了Meta信息。<template>元素则可以用于所有卡片共享的元素。例如，<template>元素可以指定事件绑定器——如对卡片集中所有卡片都使用的 do和onevent（不久我们就会遇到这种情况）。在一个卡片中定义的元素可以被指定优先级，使其优先级高于同一类型元素的优先级。

<card>元素是WML中最重要的元素。卡片中包含了确定页面内容的文本和输入元素。根据屏幕的大小不同，每个卡片或者在一个屏幕上显示，或者在较小的屏幕上分成几个小部分显示。下面的卡片包含两个表格域，客户可以根据其使用的设备功能强弱选择在一个屏幕上显示两个表格域或在两个屏幕上各显示一个表格域：

程序清单 14-4

```
<card id="author" title="Author search" newcontext="true" >
  <p mode="nowrap">
    <fieldset>
      First name:
      <input name="fname" type="text" title="Enter name"/>
    </fieldset>
    <fieldset>
      Last name:
      <input name="lname" type="text" title="Enter name" />
    </fieldset>
    <anchor>Done
      <go method="post" href="search">
        <postfield name="fname" value="$fname" />
        <postfield name="lname" value="$lname" />
      </go>
    </anchor>
  </p>
</card>
```

每个<card>元素用一个id号识别，从而使用户可以在卡片集中的卡片之间浏览。当 URL请求浏览一个卡片集时，卡片的id号被看作是片段标志（fragment identifier）。例如，http://www.wrox.com/wap/main.wml#author将显示带有作者标志的卡片。如果一个文件中包含一个以上的卡片并且没有片段标志，用户代理（应用程序）将会最先处理第一个卡片。

卡片的通常使用方法是：一般第一张卡片用做主卡，其余的卡片包含一些帮助信息或用户发出请求时显示的子菜单。在HTML网页上，所有这些信息都在同一页面上显示，如果分割成许多页面则会导致额外的网络数据传输。卡片集实际上是一个传递众多小网页——卡片的存储器。

<card>元素的title属性指明了卡片的标题。一些浏览器显示标题的方式同 HTML浏览器显示<title>元素的方法相似——在页面的上部显示。但也有些浏览器处理 title属性的方法和HTML处理title属性一样——将title属性作为工具条或元素标题。但不管怎样你应该使用 title属性，因为不管它以何种方式显示，它对用户都有帮助。上面的程序的显示将如图 14-4所示（具体的显示形式随电话型号而变化）。

在显示屏幕小的电话上，浏览器将一个卡片分成几个叫做域的小部分。域没有相应的元素表示，但 WML规范中指定如果有 <input> 或 <select> 元素，那么域就由这个元素前面的字符和元素本身组成。为了保证浏览器对卡片的分割位置恰当，使用 <fieldset> 元素设定必须一起显示的文字。ordered 属性定义了卡片中各个域的相互关系，并且可以用于在浏览器中显示卡片内容（但大部分浏览器忽略了这个属性）。

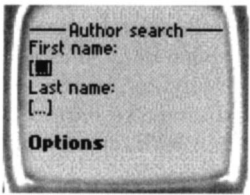


图 14-4

表14-1列出了文档结构中的元素和属性。

表 14-1

元 素	是否必须有	属 性
<wml>	必须有	id class xml:lang
<head>	可选	id class
<template>	可选	id class onenterforward onenterbackward ontimer
<card>	必须有	id class xml:lang title newcontext = (true false) ordered = (true false) onenterforward onenterbackward ontimer

14.3.3 通用属性

同XML的其他许多应用一样，WML有一些在大多数元素上都定义的通用属性。在 WML中，这些属性是id、class和xml:lang。id属性是XML ID的一种属性类型，为卡片集中每个元素分配一个唯一的名称。class属性将一个元素连接到一个或多个等级上。这样，多个元素可以有一个相同的等级名称，卡片集中享有相同等级名称的所有元素可以看作是属于同一等级。这一点同HTML是一样的，并且等级可以将样式表信息挂在文件结构上——虽然WAP规范中没有定义样式表的部分。不管怎样，一个 WAP浏览器都可以支持样式表，但当前大部分移动电话由于屏幕太小而无法支持样式表。因此，class属性的特性只能以后再使用了。xml:lang则是一个众所周知的XML属性。下面的章节中对这些通用属性就不再详述了。

14.3.4 WML包括什么

首先，我们来看一看无线标记语言都有什么功能：

- meta信息。
- 基本的文字、表格和表示。
- 事件处理器。
- 变量。
- 任务和菜单。
- 模板。
- 表单数据。
- 图像。

对HTML开发者来说，meta信息、文本、表格、表示、表单和图像都是他们所熟悉的概念。HTML中的<a>元素用于超级链接，<p>元素用于格式化文本，实现同一功能的还有最通用的表单元素<input>和<select>。

新的功能包括用变量实现一个简单的客户端模板机制，以及在卡片之间保持状态。模板是一种文件，但它可以在同一种格式下使用不同的数据。由于状态是用于不同卡片之间的，一张卡片可以显示同一卡片集中前一张卡片上用户输入的信息。这种方式有助于你在数据传送给服务器之前确认用户输入的信息。通用事件处理器、任务和计时器也是HTML中没有的新功能，但将来在W3C建议标准中将会有相似的并且功能更强的特性。

14.3.5 Meta信息

WML文件的meta信息通过< meta >元素定义。这一点也同HTML相同，并且其使用也像在HTML中一样，要慎重。浏览器可以随意忽略meta信息，所以除非你知道一个特定的浏览器会接受你输入到元素中的信息，否则你写入的信息除了使文档更加杂乱以外毫无用处。在HTML中<meta>元素同其他一些元素一起确定文件的字符编码，但我们都知道WML是XML的一个应用，因此，XML的编码规范就用于WML。

另外一个元素——<access>元素是用来对页面访问进行限制的。表14-2列出了meta信息的元素。

HTML中< meta >元素的一个非标准用法是经过一段指定的时间（根据refresh指令）后刷新页面，但在WML中刷新只能用<timer>元素（在事件部分将会看到）。

forua属性是中继代理（如WAP网关）的处理指示，在HTML的< meta >元素中没有这个属性。一些指示只是针对网关的。如果这个属性的值为“false”，则缺省的中继代

表 14-2

元素	属 性
< meta >	id
	class
	http-equiv
	name
	forua = (true false) "false"
	content
<access>	scheme
	id
	class
	domain
	path

理就会去掉< meta >元素，不向客户传送< meta >元素。一些浏览器厂商用带名称和内容属性的

< meta >元素完成一些特定的功能。

<access>元素可以用来限制对文件的访问。如果不指定 <access>元素，那么每个人都可以访问文件。如果指定 <access>元素，浏览器将只处理指定 URL（发出访问请求的连接元素的 URL）与<access>元素中domain和path属性相匹配的文档。与指定URL的域名部分后缀匹配的是访问域，与指定URL路径部分前缀匹配的是访问路径。例如，如果访问控制属性是：

程序清单 14-5

```
domain="wrox.com"
path="/books"
```

则指定URL http://wrox.com/books/xml属于卡片集，但URL http://www.wapforum.org/books不属于卡片集。<access>元素并不是安全机制（如 HTTP 认证）的一个替代项。但它可以对那些初始状态明确（例如一些变量值确定）的文档进行访问限制。

14.3.6 基本字符、表格和演示

WML中所有基本的字符和页面元素都出自于 HTML。多数情况下元素的属性都去掉了，表 14-3列出了基本字符和页面的元素。

表 14-3

元 素	属 性
链接	
<a>	id class xml:lang href title
表格	
<table>	id class xml:lang title align columns
<tr>	id class
<td>	id class
文本结构	
<p>	id class xml:lang align mode

(续)

元 素	属 性
 	id class
表示	
	id class
<u>	id class
<I>	id class
<big>	id class
<small>	id class
段落	
	id class
	id class

表14-3中的大部分元素大家都很熟悉，所以我们将只讨论那些和 HTML不同的元素。值得注意的是元素<table>，它的属性align和columns是WML所特有的。属性 align用于指定表格中各列的排列方法。属性 columns必不可少，用来指定表格中列的数目。属性 align的值是一串字母，每个字母代表一列（参见表 14-4）。

例如，“LLR”表示头两列应该左对齐，后一列应该右对齐。

元素<p>的属性mode是WML特有的。当段落中的文字与电话的显示屏幕不相匹配时，属性 mode 指定文字显示的方式。mode 值为“nowrap”，一些浏览器会采用不换行水平滚动显示的方式显示文字；如果 mode 值为“wrap”，文字会以HTML的通常段落格式显示，并且在需要时换行。元素<p>的属性align的值为left、right和center，用来指定段落中文字排列的方式（和元素<table>不同）。

WML中有表示元素如、<i>和<u>，但没有标题元素（如HTML中的元素<h>）。所以你能只能用表示元素或段落元素来指定标题。虽然现在大部分带 WAP的移动电话都可以显示黑体和斜体文字，并且今后更多的移动电话都会带有这个功能，但你也不要依靠浏览器去实现这些元素的功能。另外你也不要表示元素传递表达信息。例如，如果用 <big>作为一个标题的格式，若浏览器不能显示大的字符，它完全可能忽略这个元素，也不知道这些字是标题。同样，也要避免使用下划线文字，因为在一些浏览器中下划线很像是一个链接。

浏览

当你为小型产品开发应用程序时，浏览是最难令人满意的事情之一。下面是一些有用的提

表 14-4

align的值	含 义
“ L ”	左边
“ R ”	右边
“ C ”	居中

示：

- 用元素<a>和
生成一个浏览选择列表。如果选项超过 9个，用“More...”链接到一个新的列表。
- 在每一张卡片里要确保用户可以随时浏览上一页、下一页或者回到一个人都熟悉的起始页面。例如，在一个复杂的菜单层次中让用户可以随时回到“主菜单”。
- 为用户提供回到熟悉站点的快捷方式。避免用户为了返回到“主菜单”，要点击10次“返回”键的情况发生。
- 对链接使用短名称，避免太多的空格。

下面是一个浏览选项列表的例子，大多数应用程序包括的卡片类型有：

程序清单 14-6

```
<card id="main" title="WROX">
  <do type="prev" label="Back">
    <prev/>
  </do>
  <p mode="nowrap">
    <a href="#author">Search Author</a><br/>
    <a href="#title">Search Title</a><br/>
    <a href="#about">About</a><br/>
    <anchor>Done
      <go href="http://www.wrox.com/wap/index.wml" />
    </anchor>
  </p>
</card>
```

14.3.7 使用计时器

计时器可以用来帮助你在卡片之间浏览。元素 <timer>可以在一段指定时间后生成一个ontimer事件。元素<timer>的结构如下：

程序清单 14-7

```
<timer name="V" value="Time" />
```

属性name指定变量的名称。计时器设定为变量 V的值。如果没有指定的变量或者指定变量没有数值，计时器设定为数值属性 time的值。时间以十分之一秒为单位。

元素<timer>的理想用法是在服务启动之前短时间显示一张卡片或文档上的商业信息。例如，如果用户想要链接一个开放餐馆的名单列表，最先显示的卡片是一个餐馆的标识，几秒钟后，显示餐馆列表的卡片，这个过程同运行程序的屏幕显示一样，只不过现在是用于给产品或服务做广告。元素<timer>的其他应用例子包括：

- 每隔一段时间显示一条简短的帮助信息。移动电话上这个功能很普遍，如果用户使用停止一段时间，帮助信息就会自动显示。
- 引导用户穿越一系列卡片，如一个长的新闻故事。如果用户没有任何动作（没有按任何键），服务将会自动调用下一张卡片，也就是说我们假定用户会选择下一张卡片。这种方式产生

了一种有效的“不需用手”的浏览服务。

14.3.8 事件处理器

为了捕捉一个 ontimer 事件，要使用事件处理器属性 ontimer 或元素 <onevent>。属性 ontimer 在卡片元素中定义。属性的值可以是一个 URL 或一个片段标识。下例中 ontimer 事件将会激发下一张卡片的加载：

程序清单 14-8

```
<card id="about" title="WROX" ontimer="#about2">
  <timer value="100"/>
  <p mode="nowrap">
    <a href="#main">Done</a><br/>
    Wrox Press aims to make programmers successful by sharing with them the
    knowledge and experience of their professional peers.
  </p>
  <p>
    <a href="#main">Done</a>
    <a href="#about2">More</a>
  </p>
</card>

<card id="about2" title="WROX">
  <p mode="nowrap">
    <a href="#main">Home</a><br/>
    <a href="allbooks.wml">All Books</a><br/>
    <a href="http://www.wroxconferences.com">Conferences</a><br/>
    <a href="allauthors.wml">Authors</a><br/>
    <a href="support.wml">Support</a><br/>
    <a href="membership.wml">Membership</a><br/>
    <a href="contacts.wml">Contact Us</a><br/>
  </p>
</card>
```

在这个例子中，第一张卡片激活后约 10 秒钟生成 ontimer 事件。许多开发者都发现在卡片元素中使用事件处理器属性 ontimer 十分方便。但你也可以用元素 <onevent> 去捕获任何一个有名称的事件。在下面的例子中元素 <onevent> 位于元素 <card> 中，用来捕获一个 ontimer 事件：

程序清单 14-9

```
...
<onevent type="ontimer">
  <go href="#about2" />
</onevent>
<timer value="100" />
...
```

可以用 <onevent> 元素指定事件绑定，或者将事件绑定作为事件处理器的一个属性（参见表 14-5）。这些可以在元素 <card> 和元素 <template> 中指定。元素 <card> 中的事件绑定优先权总是高于元素 <template> 中同一类型的事件绑定。元素 <do> 也有类似的“遮盖”规则，我们在以后的章节中将会看到。

表 14-5

事 件 元 素	属 性	事 件 元 素	属 性
<timer>	id	<onevent>	id
	class		class
	name		type
	value		

在14.3.10节“任务和菜单”中，我们还将提到 onevent事件，onevent事件的主要用途就是将任务和一个特定类型的事件绑定在一起。

14.3.9 使用变量

变量可以用来在几个文件之间保存状态信息。一个变量就是一个具有数值的名称。元素<setvar>用来将数值赋予一个变量：

程序清单 14-10

```
<setvar name="N" value="V" />
```

属性name确定变量名称，属性 value确定变量的值。元素<setvar>可以在元素<onevent>和元素<do>里使用。而且，变量名称通过元素<input>和元素<select>和一个值绑定在一起。

为了在内容中包括变量的值，采用了符号 \$varname或\$(varname)（后者更常用）。

程序清单 14-11

```
The timer is $(Timer).
```

一个变量的取值范围称为上下文。元素<card>上的newctx属性可以产生一个新的上下文。当生成一个新的上下文时，所有的变量值和历史列表被清除了。所以你可以利用 newctx属性（像上例一样）来保证前面服务所使用的一些旧变量不会和你的变量发生冲突。

下面是一个变量如何在卡片之间保持状态的例子：

程序清单 14-12

```
<wml>
  <card id="card1" title="$title" >
    <onevent type="onenterforward">
      <refresh>
        <setvar name="title" value="Variables" />
      </refresh>
    </onevent>
    <onevent type="onenterbackward">
      <refresh>
        <setvar name="title" value="Try again" />
      </refresh>
    </onevent>
  <p>
    Name <input type="text" name="fname" title="Enter name" />
```

```

        <anchor>Done
          <go href="#card2"/>
        </anchor>
      </p>
    </card>

    <card id="card2" title="Variables" >
      <p>
        Your name is ${fname}?
        <anchor>Yes
          <go href="next.wml" />
        </anchor>
        <anchor>No
          <prev />
        </anchor>
      </p>
    </card>
  </wml>

```

第一张卡片的标题是一个变量——title。当用户进入第一张卡片时，变量 title 设定为“Variables”。这时要求用户输入名称，输入的名称存储在变量 fname 中。当用户浏览到第二张卡片时，变量 fname 包含在内容中，用户在这里对在上一张卡片里输入的名称进行确认。如果用户选择 no，浏览器将会再次显示第一张卡片。在从第二张卡片返回到第一张卡片的过程中，元素 <prev> 产生了一个 onenterbackward 事件，这个事件被第一张卡片捕获后，会将变量 title 设定为“Try again”。图 14-5 显示了这个过程。

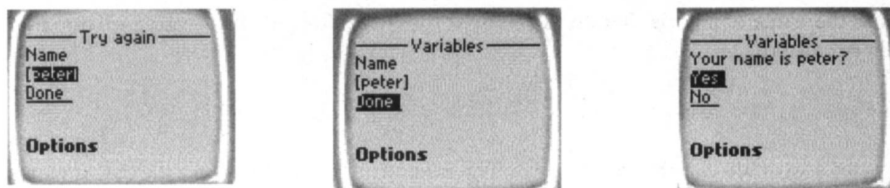


图 14-5

既然变量能在卡片之间保存状态，我们可以用变量在数据传送给服务器之前让用户对数据确认，同一张卡片可以在不同的用户界面上重复使用，而无须任何脚本或额外的网络访问。

下例中用元素 <setvar> 来设定计时器的超时。计时器的值存储在变量 T 中，计时器的当前状态存储在变量 Timer 中。当用户进入卡片中、计时器超时和用户选择 Restart 时，变量就会发生改变：

程序清单 14-13

```

<card title="Timer">
  <onevent type="onenterforward">
    <refresh>
      <setvar name="Timer" value="Running" />
      <setvar name="T" value="10" />
    </refresh>
  </onevent>
  <onevent type="ontimer">

```

```
<refresh>
  <setvar name="Timer" value="Timeout" />
  <setvar name="T" value="0" />
</refresh>
</onevent>
<timer name="T" value="10" />
<do label="Restart" type="accept" >
  <refresh>
    <setvar name="Timer" value="Running" />
    <setvar name="T" value="10" />
  </refresh>
</do>
<p>
The timer is $(Timer).
</p>
</card>
```

变量名称对大小写敏感，由US-ASCII码字母或下划线后加0、字母、数字或下划线组成，变量的值看作是CDATA（因此你不能在变量中带有标记）。

变量可以在包含明文（PCDATA）的元素中使用，也可以在下列属性值中使用：

- 所有的title属性。
- 所有的value和ivalue属性。
- 所有的href和src属性。
- 元素<do>上的label属性。
- 元素<postfield>和<setvar>上的name属性。
- 元素上的alt和localsrc属性。
- 事件处理器属性onpick、ontimer、onenterforward和onenterbackward。

通过在变量名称上加一个指令，可以指定变量值为不带URL的值、带URL的值或者是当变量值包含在内容里时变量值保持不变。

- 变量值保持不变\$(varname:noesc)。
- 不带URL的值\$(varname:escape)。
- 带URL的值\$(varname:unesec)。

由于字母\$代表变量，所以在字符数据和属性值里使用它时，“\$”会被忽略掉。“\$\$”符号用于忽略一个句子，在显示数据或处理属性之前用一个“\$”符号代替“\$\$”：

程序清单 14-14

```
This is a dollar sign $$.  
This is the value of the N variable $(N)
```

14.3.10 任务和菜单

在WML中开发者对浏览器的软键和菜单有一定程度的控制。软键控制是一种程序控制，使浏览器或者显示一个单独的菜单、一个单独的键，或者浏览器菜单的一部分。通过元素<do>，一些任务如浏览和更新变量可以设定到浏览器上提供的软键和菜单上。也可以通过使用元素<anchor>使任务在线连接。表14-6中的元素可以生成菜单和任务。

表 14-6

任 务 元 素	属 性
<anchor>	id class title
<do>	id class type label name optional = (true false) "false"
<go>	id class href sendreferer method = (post get) "get" accept-charset
<prev>	id class
<refresh>	id class
<noop>	id class

1. 生成菜单和软键——元素<do>

内容开发者可以调用元素 <do>指定与卡片连接的菜单。至于菜单的显示方式则取决于浏览器。一些浏览器产生一个带有菜单和选项的链接将元素 <do>和内容连接起来，而另一些浏览器则以弹出框和按钮列表形式显示菜单。

元素<do>代表一个任务，属性 label确定其名称。一般情况下建议使用少于 12个字符的短名称，并且避免使用空格。一旦选定元素，就会执行相应的任务。元素 <do>同元素<onevent>一样，可以包含任务元素<go>、<prev>、<refresh>和<noop>。

下面的例子生成了一个包括两项的菜单：

程序清单 14-15

```
<card id="form" title="Author search" newcontext="true">
  <do type="prev" label="Back">
    <prev/>
  </do>
  <do type="accept" label="Done" >
    <go method="post" href="search">
      <postfield name="type" value="bio" />
      <postfield name="fname" value="$fname" />
      <postfield name="lname" value="$lname" />
    </go>
  </do>
</card>
```

```

</do>
<p>
<fieldset>
    First name:
    <input name="fname" type="text" title="Enter name"/>
</fieldset>
<fieldset>
    Last name:
    <input name="lname" type="text" title="Enter name" />
</fieldset>
</p>
</card>

```

参照下面章节中对元素<go>和<prev>的说明。

图14-6是Search表单的屏幕拷贝。

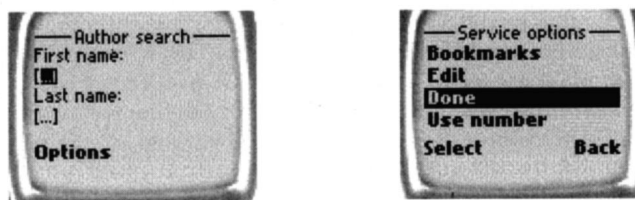


图 14-6

元素<input>和<fieldset>与HTML中的一样，我们将在下面更详细地讨论这两个元素。元素<do>的属性type的目的是帮助浏览器选择一个显示元素的适当方式。任何时候你都要指定一个type，在WML的规范中有各种type的说明（参见表14-7）。

表 14-7

类型	描 述
accept	肯定确认（接受）
prev	返回到上一个浏览的页面
help	请求帮助，有可能和上下文相关
reset	清除或重设状态
options	请求与上下文相关的选项或其他操作
delete	删除子项或选项

2. 任务<go>

任务元素<go>用来使浏览器访问其他资源或将数据传送给网络服务器。它实现了在 HTML 中元素<form>完成的许多功能。下面的例子演示在用户选择了 Done菜单项时，元素<go>怎样将数据传送给网络服务器：

程序清单 14-16

```

<do type="accept" label="Done" >
    <go method="post" href="search">
        <postfield name="fname" value="$fname" />
        <postfield name="lname" value="$lname" />
    </go>
</do>

```



```
</go>
</do>
```

元素<go>中用<postfield>构成了一个HTTP的发送请求。此外还可以在元素<go>中使用一个或多个<setvar>元素（上例中没有这么做），通过这种方法就可以在目标资源中使用变量。

元素<go>一旦激活，就会激发一个 oneventforward 事件并将其放入目标资源中。

3. 任务<prev>

任务元素<prev>可以在浏览器中返回到历史浏览记录中最近的一个页面上。WML的历史记录和通常的HTML浏览器历史记录略有不同。WML的历史记录是一个旧的HTTP请求的列表（URL、方法、传输数据和标题），而不是像HTML历史记录一样，是一份显示给用户的资源拷贝。当你返回到WML的历史记录中时，系统将重新执行请求，以前调用的文档从网络或本地缓存中再次调用。下面是生成含有back键的菜单的例子：

程序清单 14-17

```
<do label="Back" type="prev">
  <prev/>
</do>
```

元素<prev>可以包含元素<setvar>。所以，完全有可能在返回浏览历史记录时使用新的变量值。

元素<prev>一旦激活，就会激发一个 onenterbackward 事件并将其放入目标资源中。

4. 任务<noop>

任务元素<noop>实际上没有什么作用，它存在的目的是为了设计一种方法使文件里<template>元素中的元素<do>可以被卡片中的元素<do>覆盖——“遮盖”。例子如下：

程序清单 14-18

```
<wml>
<template>
  <do label="Back" name="prev" type="prev">
    <prev/>
  </do>
</template>
<card>
  <do name="prev" type="prev">
    <noop/>
  </do>
  <!-- This card will not display the "Back" key -->
</card>
<card>
  <!-- This card will display the "Back" key from the template -->
</card>
</wml>
```

如果一个元素<card>中的元素<do>包含任务元素<noop>，那么这张卡片中的元素<do>将会覆盖文件中<template>元素里任何具有相同名字的元素<do>。由于元素<noop>并没有做任何事情，其效果仅仅是使模板级的元素<do>被卡片级的元素<do>禁用。因此，任务<noop>可以同样

用于带有元素 `<onevent>` 的事件绑定。

5. 任务 `<refresh>`

任务元素 `<refresh>` 提供了一种刷新变量值的机制。在下例中，一旦选择了 Restart 选项，变量 T 和 Timer 将被赋予新的数值：

程序清单 14-19

```
<do label="Restart" type="accept" >
<refresh>
  <setvar name="Timer" value="Running" />
  <setvar name="T" value="10" />
</refresh>
</do>
```

元素 `<do>` 可以包含在元素 `<card>` 或 `<template>` 中。当处理卡片时，模板中的元素 `<do>` 和卡片中的元素 `<do>` 连接起来。除非是像我们上面描述的那样元素 `<do>` 被一个 `<noop>` 任务所覆盖。

6. 访问队列中的任务——元素 `<anchor>`

由于元素 `<do>` 的显示方式是由浏览器决定的——一些浏览器以软键显示，另一些则以菜单形式显示——所以在不同的浏览器上实现同样的显示服务很困难。作为一种替代方法，或是补充方法，元素 `<anchor>` 用于在内容中包括一个用做链接的任务。

下例中元素 `<anchor>` 用来将数据传送给服务器：

程序清单 14-20

```
<card id="author" title="Author search" newcontext="true" >
  <p mode="nowrap">
    <fieldset>
      First name:
      <input name="fname" type="text" title="Enter name" />
    </fieldset>
    <fieldset>
      Last name:
      <input name="lname" type="text" title="Enter name" />
    </fieldset>
    <anchor>Done
      <go method="post" href="search">
        <postfield name="type" value="bio" />
        <postfield name="fname" value="$fname" />
        <postfield name="lname" value="$lname" />
      </go>
    </anchor>
  </p>
</card>
```

在上面的卡片中使用元素 `<anchor>` 而不使用元素 `<do>` 的好处在于：作为一个开发者，你很清楚在卡片的结尾将会显示 Done 选项。如果使用元素 `<do>`，Done 选项的显示是由浏览器决定的，因此在不同产品上的显示结果不一样。为了保险起见，你可以同时使用元素 `<do>` 和元素 `<anchor>`。

14.3.11 客户端模板

通过变量，WML 文档可以作为电话中的本地模板。这种方法同许多网络开发者的做法相似

——当动态生成文档时使用模板。模板可以在产品中存储，也可以供多种服务使用。例如，在查找服务中，从服务器返回的数据显示文档有可能是一个模板。模板文档包括所有和结果一起显示给用户的信息，如结构、显示方式、链接以及帮助。这样，服务器返回的卡片只需要包括有用数据就行了，这就是模板的工作原理。

首先，我们生成查询表单卡片——author，卡片中包括全部结构、显示方法以及所有我们希望用户在显示结果中看到的额外链接。注意这个卡片集还包括用于显示结果的卡片 bio：

程序清单 14-21

```
<wml>
...
<card id="author" title="Author search" newcontext="true" >
  <p mode="nowrap">
    <fieldset>
      First name:
      <input name="fname" type="text" title="Enter name" />
    </fieldset>
    <fieldset>
      Last name:
      <input name="lname" type="text" title="Enter name" />
    </fieldset>
    <anchor>Done
      <go method="post" href="search">
        <postfield name="fname" value="$fname" />
        <postfield name="lname" value="$lname" />
      </go>
    </anchor>
  </p>
</card>

<card id="bio" title="Biography">
  <p mode="nowrap">
    <a href="#main" >Done</a>
    $(bio)
    <a href="#main" >Done</a>
  </p>
</card>
...
</wml>
```

这个模板卡片包括一个变量——bio。当显示卡片 bio 时，我们希望用服务器返回的数据代替变量名称。

服务器查询到请求的数据后将下面这张卡片发送回用户端，这张卡片中有查询结果，但不带有文本结构和显示信息：

程序清单 14-22

```
<card newcontext="true">
  <onevent type="onenterforward">
    <go href="main.wml#bio">
      <setvar name="bio" value="Peter Stark works as architect at Phone.com,
        Redwood City, Ca, and represents his company in the WAP Forum." />
    </go>
```

```
</onevent>
</card>
```

上面这张卡片的作用好像是一个“蹦床”。当用户代理访问这张卡片时，对其进行变量值设定，然后处理过程转移到模板卡片——bio上。这也就是说既然通过空中从服务器传回来的卡片中只含有数据，那么它的尺寸就会最小，因为所有的格式信息都存储在客户端。onenterforward事件可以被指向这张卡片的链接元素激活。我们稍后再回到事件和任务上。

当处理卡片时，变量bio被赋予数值，那么卡片bio将变成：

程序清单 14-23

```
<card id="bio" title="Biography">
  <p mode="nowrap">
    <a href="#main" >Done</a>
    Peter Stark works as architect at Phone.com, Redwood City, Ca, and
    represents his company in the WAP Forum.
    <a href="#main" >Done</a>
  </p>
</card>
```

需要注意的是上面提到的“蹦床”卡片并不显示任何内容。但不管怎样这张卡片要保留在历史记录中。如果用户返回浏览就会出现一个问题：显示一张空白卡片。解决这个问题的办法是在紧跟空白卡片的卡片中不要使用任务<prev>——即Back控件。

14.3.12 表单数据

HTML中的表单元素<input>、<select>、<option>、<optgroup>和<fieldset>在WML中也有，并且用途和HTML中的一样。但它们并没有保留所有的HTML属性，此外还增加了一些属性。我们已经讨论了元素<go>和元素<postfield>如何向服务器传输数据，其中元素<go>代替了HTML中元素<form>的功能。这一部分将讨论HTML表单和WML表单的差异。首先，WML的表单元素及其属性见表14-8。

表 14-8

表单域 元素	属 性
<select>	title
	name
	value
	iname
	ivalue
	multiple = (true false) "false"
	tabindex
<input>	name
	type = (text password) "text"
	value
	format

(续)

表单域 元素	属 性
	emptyok = (true false) "false"
	size
	maxlength
	tabindex
	title
<option>	value
	title
	onpick
<optgroup>	title
<fieldset>	title

1. 元素<input>

为了加强输入确认，在 HTML 的元素<input>基础上增加了两个属性。属性 emptyok 指定当数据传输给服务器时某个输入元素是否必须键入数值或可以为空。下面是在数据传给服务器前要求用户输入名称的一个示例：

程序清单 14-24

```
Name<input name="N" emptyok="false" type="text" title="Enter name" />
```

如果没有属性 emptyok，服务器就不得不检查输入元素是否输入了数据。通过使用属性 emptyok，可以在客户端进行检验，从而节约网络访问时间。

属性 format 确定用户想要的格式。在下例中要求用户键入电话号码，格式为 12345-123：

程序清单 14-25

```
Name<input name="N" format="NNNNN\ -3N" type="text" title="Enter name" />
```

WML 规范中定义了属性 format 的字符值，这里列出表 14-9 以供参考。

表 14-9

值	描 述
A	输入大写字母或标点符号（非数字的大写字符）
a	输入小写字母或标点符号（非数字的小写字符）
N	输入数字符号
X	输入大写字母
x	输入小写字母
M	输入任何字符，为了输入简单用户可以设定所有的字符都是大写的，但输入时允许输入任何字符
m	输入任何字符，为了输入简单用户可以设定所有的字符都是小写的，但输入时允许输入任何字符
*f	输入数字符号：f 是包含在上述格式码中的一个，可以指定输入的符号类型。注意：这种格式只能使用一次并且必须是位于格式串的末尾

(续)

值	描 述
nf	输入n个字符, n的值为1~9, f是上述格式符号中的一种(除去* f格式符号), 并且指定可以输入的符号类型。注意: 这个格式只可以指定一次并且必须在格式字符串的末尾显示
\c	显示输入域中的下一个字符——c; 允许忽略格式字符, 也允许引入非格式化字符, 使它们可以在输入域中显示。忽略的字符看作是输入值的一部分, 并且必须由用户代理保存。例如, 掩码为“ NNNNN\3N ”的输入域“ 12345-123 ”的存储值是“ 12345-123 ”, 而不是“ 12345123 ”。同样, 如果名称属性确定的变量值是“ 12345123 ”, 掩码是“ NNNNN\3N ”, 那么用户代理必须取消变量设定, 因为它与掩码不符

2. 元素<select>

WML中采用的另外一个 HTML 表单域是元素<select>。元素<select>在WML中的功能和在HTML中的相似, 并且可以包括元素<option>和<optgroup>。

属性onpick可以使元素<option>成为一个链接元素, 当用户选择这个选项时浏览器就会进入到属性所指定的 URL 上。但是, 你可能并不想这么做, 因为用户会对这种方式感到惊讶(他们认为选项列表仅仅是一个选项列表, 而不是一个链接列表), 并且一些浏览器显示元素<select>的方式使其不利于使用浏览器选项。

下例中我们使用元素<select>实现用户选择图书目录的功能:

程序清单 14-26

```
<card id="title" title="Title search" newcontext="true" >
  <p mode="nowrap">
    <fieldset>
      Word in title:
      <input name="title" type="text" title="Enter title"/>
    </fieldset>
    <fieldset>
      Category:
      <select name="cat" title="Select cat.">
        <option value="none">-none-</option>
        <option value="asp">ASP</option>
        <option value="cpp">C++</option>
        <option value="com">COM</option>
        <option value="dbs">Database</option>
        <option value="gnu">GNU/Linux</option>
        <option value="java">Java</option>
        <option value="vb">VB</option>
        <option value="xml">XML</option>
      </select>
    </fieldset>
    <anchor>Done
      <go method="post" href="search">
        <postfield name="type" value="books" />
        <postfield name="title" value="$title" />
        <postfield name="cat" value="$cat" />
      </go>
    </anchor>
  </p>
</card>
```

注意上例中我们使用了一个 none 选项，以备用户不想选择任何分类目录时用。

表单在无线应用中十分重要。几乎每一项应用都遵循同一模式：用户请求访问某些信息，然后从数据库中向用户返回结果。移动电话的大部分应用程序包括动态生成内容，很少有像传统 Web 上那样的静态文档如主页。

14.3.13 图像

虽然你并不打算在移动电话上使用图像，但大部分 WAP 浏览器至少支持简单的 WAP 点位图像——WBMP。WBMP 的规范可以参阅 WAP 论坛站点——<http://www.wapforum.org/>，但你在查找过程可能要费点力气，因为这个规范只有两页，并且包含在一个叫 WAE 规范的文档中。一些浏览器除了支持 WBMP 之外，还支持标准网络图像格式如 GIF。

为了在文档中包含图像，使用一个类似于 HTML 的元素 。但对其增加了一个 WML 特征属性——localsrc 属性。这个属性允许使用一个本地图像，但不能从网络服务器上调用图像。本地图像的名称还没有在 WAP 中给定，所以目前这些图像还是专用图像，当然在写本书时为本地图像指定名称的工作已经开始进行了。目前，要使用 localsrc 属性，你必须先清楚每个 WAP 浏览器厂家支持什么格式的图像。在下面的例子中，如果本地有图像 book，则调用本地图像，否则将从网络服务器上调用图像 book.wbmp。使用本地图像的好处是浏览器不必到网上查找，因而节约网络时间，并且图像也可以优化，使其与设备的用户界面的相容性更好。

程序清单 14-27

```

```

其他的图像属性如 alt、src、vspace、hspace、align、height 和 width 同 HTML 中的一样。表 14-10 是为图像定义的元素和属性。

表 14-10

图像元素	属 性
	alt
	src
	localsrc
	vspace
	hspace
	align
	height
	width

14.4 在服务器上生成 WML

许多原因决定了小型产品上提供的服务一般是动态的。因此，你不能写一个静态的 WML 文档，而是通常要建立动态的 WML 卡片。

当然，我们可以使用任何一种可以生成动态 HTML或XML的技术，如ASP、JSP、CGI脚本等。但一个理想的生成 WML文档的工具是XSL转换（XSLT）语言。

在服务器上我们可以用 XSLT将任何XML源文档转换成 WML文档。下面的XML文档是从数据库返回的查询结果：

程序清单 14-28

```
<Author>
  <FirstName>Peter</FirstName>
  <MI>M</MI>
  <LastName>Stark</LastName>
  <Biographical>
    Peter Stark works as architect at Phone.com, Redwood City, Ca, and
    represents his company in the WAP Forum.
  </Biographical>
  <Portrait type="image/jpeg" href="stark.jpg"/>
  <Portrait type="image/vnd.wap.wbmp" href="stark.wbmp"/>
</Author>
```

我们假定移动电话上已经有文档 main.wml——否则下载这个文档并将其放入到本地缓存中——我们通过空中传输的只是一个设定变量 bio的最小化文档。在演示 onenterforward事件的应用时已经用过这个文档。

下面是生成 WML文档的转换页：

程序清单 14-29

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="Author">
<wml>
  <card newcontext="true">
    <onevent type="onenterforward">
      <go href="main.wml#bio">
        <setvar name="bio" value="{Biographical}" />
      </go>
    </onevent>
  </card>
</wml>
</xsl:template>

<xsl:template match="@*" >
  <xsl:apply-templates />
</xsl:template>

<xsl:template match="* | text()" >
  <xsl:apply-templates />
</xsl:template>

</xsl:stylesheet>
```

图像可能已经被转换成 WBMP或GIS格式了，具体的格式则取决于电话的功能。自然这个过程需要一个可以完成图像转换的图形库。

讨论过WAP和WML的基础理论和WAP产品的标记语言后，我们来看看如何将这些应用到实践中去。

14.5 WROX的WML应用

这一部分将我们前面生成的应用程序小结一下，这些内容是 WROX网站的一个简化版。用户可以完成如下功能：

- 得到关于WROX的通常信息。
- 根据姓或名查询作者的生平。
- 根据分类和/或标题词语查询书的标题。

下面是WML代码：

程序清单 14-30

```
<wml>

<template>
  <do type="prev" label="Back">
    <prev/>
  </do>
</template>

<card id="main" title="WROX">
  <onevent type="ontimer">
    <go href="#about" />
  </onevent>
  <timer value="100" />
  <p mode="nowrap">
    <a href="#author">Search Author</a><br/>
    <a href="#title">Search Title</a><br/>
    <a href="#about">About</a><br/>
  </p>
</card>

<card id="author" title="Author search" newcontext="true" >
  <p mode="nowrap">
    <fieldset>
      First name:
      <input name="fname" type="text" title="Enter name"/>
    </fieldset>
    <fieldset>
      Last name:
      <input name="lname" type="text" title="Enter name" />
    </fieldset>
    <anchor>Done
      <go method="post" href="search">
        <postfield name="type" value="bio" />
        <postfield name="fname" value="$fname" />
        <postfield name="lname" value="$lname" />
      </go>
    </anchor>
  </p>
</card>
```

```

<card id="bio" title="Biography">
  <do type="prev" >
    <noop/>
  </do>

  <p mode="nowrap">
    <a href="#main" >Done</a>
    $(bio)
    <a href="#main" >Done</a>
  </p>
</card>

<card id="title" title="Title search" newcontext="true" >
  <p mode="nowrap">
    <fieldset>
      Word in title:
      <input name="title" type="text" title="Enter title"/>
    </fieldset>
    <fieldset>
      Category:
      <select name="cat" title="Select cat.">
        <option value="none">-none-</option>
        <option value="asp">ASP</option>
        <option value="cpp">C++</option>
        <option value="com">COM</option>
        <option value="dbs">Database</option>
        <option value="gnu">GNU/Linux</option>
        <option value="java">Java</option>
        <option value="vb">VB</option>
        <option value="xml">XML</option>
      </select>
    </fieldset>
    <anchor>Done
    <go method="post" href="search">
      <postfield name="type" value="books" />
      <postfield name="title" value="$title" />
      <postfield name="cat" value="$cat" />
    </go>
  </anchor>
</p>
</card>

<card id="book" title="Book">
  <p mode="nowrap">
    <a href="#main" >Done</a>
    $(book)
    <a href="#main" >Done</a>
  </p>
</card>

<card id="about" title="WROX" ontimer="#about2">
  <timer value="100"/>
  <p mode="nowrap">
    <a href="#main">Done</a><br/>

```

Wrox Press aims to make programmers successful by sharing with them the knowledge and experience of their professional peers.

```

</p>
<p>
  <a href="#main">Done</a>
  <a href="#about2">More</a>
</p>
</card>

<card id="about2" title="WROX">
  <p mode="nowrap">
    <a href="#main">Home</a><br/>
    <a href="allbooks.wml">All Books</a><br/>
    <a href="http://www.wroxconferences.com">Conferences</a><br/>
    <a href="allauthors.wml">Authors</a><br/>
    <a href="support.wml">Support</a><br/>
    <a href="membership.wml">Membership</a><br/>
    <a href="contacts.wml">Contact Us</a><br/>
  </p>
</card>

</wml>

```

当用户查询作者时，服务器会返回一个简短的作者生平并显示在卡片 bio 中，这一点我们前面已经讨论过。在这张卡片中禁用 Back，因为我们不想让用户返回到一张空白卡片上——“蹦床”卡片，相关内容可以参阅 14.3.11 节“客户端模板”。为防止作者生平过长，我们在卡片开始和结尾处各使用了一个 Done 标识。

如果用户在第一张卡片上处于休眠状态，应用程序将自动跳到卡片 about 上，并向用户显示一些关于 WROX 的商业信息。

当用户要寻找一本书并键入标题中的词语时，服务器将会反馈回一个匹配列表。例如：

程序清单 14-31

```

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1/EN"
  "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<template>
  <do type="prev" label="Back">
    <prev/>
  </do>
</template>

<card id="first" title="XML/Scripting">
  <p mode="nowrap">

    <a href="Details.asp?ISBN=1861002718">VBScript Programmer&#39;s
      Reference</a><br/>
    <a href="Details.asp?ISBN=1861002645">Professional Visual Interdev 6
      Programming</a><br/>
    <a href="Details.asp?ISBN=186100270X">Professional JavaScript</a><br/>
    <a href="Details.asp?ISBN=1861001746">IE5 Dynamic HTML Programmer&#39;s
      Reference</a><br/>
    <a href="Details.asp?ISBN=1861002270">Designing Distributed Applications with
      XML, ASP, IE5, LDAP and MSMQ</a><br/>
    <a href="Details.asp?ISBN=1861001576">XML in IE5 Programmer&#39;s
      Reference</a><br/>

```



```
<a href="Details.asp?ISBN=1861002289">Professional XML Design and  
Implementation</a><br/>  
<a href="#second">More...</a>  
  
<anchor>Done  
  <go href="http://www.wrox.com/wap/index.wml" />  
</anchor>  
  
</p>  
</card>  
  
<card id="second" title="XML/Scripting">  
  <p mode="nowrap">  
  
    <a href="Details.asp?ISBN=1861002211">Implementing LDAP</a><br/>  
    <a href="Details.asp?ISBN=1861001525">XML Applications</a><br/>  
    <a href="Details.asp?ISBN=1861001894">JavaScript Objects</a><br/>  
    <a href="Details.asp?ISBN=1861001657">Professional Style Sheets for HTML and  
XML</a><br/>  
    <a href="Details.asp?ISBN=186100138X">Instant DHTML Scriptlets</a><br/>  
    <a href="Details.asp?ISBN=1861001274">Instant JavaScript</a><br/>  
    <a href="Details.asp?ISBN=1861001568">Instant HTML Programmer&#39;s Reference,  
HTML 4.0 Edition</a><br/>  
    <a href="Details.asp?ISBN=1861000707">Professional IE4 Programming</a><br/>  
    <a href="Details.asp?ISBN=1861000685">Instant IE4 Dynamic HTML Programmer&#39;s  
Reference - IE4 Edition</a><br/>  
    <a href="Details.asp?ISBN=1861001193">Instant Netscape Dynamic HTML  
Programmer&#39;s Reference NC4 Edition</a><br/>  
    <a href="Details.asp?ISBN=186100074X">Professional Web Site  
Optimization</a><br/>  
    <a href="Details.asp?ISBN=1861000766">Instant HTML Programmer&#39;s  
Reference</a><br/>  
  
    <anchor>Done  
      <go href="http://www.wrox.com/wap/index.wml" />  
    </anchor>  
  
  </p>  
</card>  
</wml>
```

这个匹配列表被拆分成两张卡片，从而便于用户浏览。注意在第一张卡片末尾处的 More... 选项。此外每一张卡片都为用户提供了返回到应用程序起始页的方法。通过使用模板，每张卡片上也都有一个 Back 键。

讨论完如何为新一代移动产品提供标记文件后，接下来看看这些客户设备使用的脚本语言。

14.6 WML 脚本

虽然 WML 允许我们为带 WAP 功能的产品标记文档，但 WAP 规范中包括一种称为 WMLScript 的脚本语言可以使网页增添更多的功能。WMLScript 从 ECMAScript 语言（也叫 JavaScript 或 JScript，也许后者人们更熟悉）演化而来，因此用过 ECMAScript 语言的人对它会很熟悉。虽然 WMLScript 不是 ECMAScript 的一个完整子集，但它是大多数网络开发者所熟悉的通用脚本语言的一个精简替代版。我们先讨论基本概念和如何调用脚本函数，然后再看看目前提供的脚本库。

14.6.1 基本概念

在这一部分中我们假定你已经知道 ECMAScript。我们不再解释基本的脚本语言结构（如 If 和 For 语句），而是重点讲述 WAP 浏览器中提供的脚本函数库以及脚本和 WML 如何一起使用。

WML 脚本程序没有内嵌于 WML 中（像 HTML 中的元素 `<script>` 一样），而是在独立文件中存储，并可以通过 URL 访问。

1. 函数

WMLScript 脚本程序是面向函数的。一个文件可以有几个函数，所以每个函数要有一个唯一的名称。要调用文件中某个特定的函数，需要用文件的 URL 和函数名称的片段标识。函数之间不能互相包含，因此不能在一个函数中定义另一个函数。虽然函数中的变量是局部变量，但经常用这些函数管理 WML 浏览器上的全局变量。

2. 变量

变量的数据类型取决于对它的最新赋值类型。可以使用操作符 `var` 来定义变量：

程序清单 14-32

```
var wap = "Wireless Application Protocol";
```

变量名称区分大小写，可以包括字母、数字和下划线字符，但不能用数字开头。如果变量有初始值（如上例所示），则初始值的数据类型就是这个变量的初始数据类型。变量也可以有局部范围。

14.6.2 字节码

脚本在传送给移动电话的途中要经过 WAP 网关，在网关上脚本被编译成字节码并以这种形式传到电话上。这种字节码同 Java 程序中编译的字节码不同，但使用字节码的概念和原理是相同的。字节码很容易理解而且需求的带宽也较小。

14.6.3 如何从 WML 上调用脚本函数

由于脚本函数可以用 URL 定位，因此可以用 WML 中的链接元素 `<go>` 或 `<a>` 调用脚本函数：

程序清单 14-33

```
<do type="accept">
  <go href="http://www.wrox.com/wmlscript#calculate($result, $1)" />
</do>
```

当 WML 浏览器执行上面的程序时，元素 `<go>` 指向一个脚本函数作为其目标资源。脚本在与 WML 卡片独立的上下文中执行，因此 WML 文档不能共享脚本中的变量。一旦脚本执行结束，WML 文档将重新控制操作。脚本没有返回值，但脚本可以管理 WML 文档中的变量（我们前面提到过）。如果目标资源是一个普通的 WML 文档，那么目标就会替换当前文档。

注意脚本函数不添加到历史记录列表中，所以你不能 `<prev>` 回到一个脚本去。

14.7 脚本库

没有配套的标准库就无法用 WMLScript 做任何事情，但 WAP 浏览器可以支持 6 个标准库：WMLBrowser、Dialogs、Lang、Float、String 和 URL。

使用脚本库可以完成下列任务：

- 通过在 WML 文档中设定和改变变量来设定 WML 浏览器的状态。
- 生成简单的弹出式对话框，请求用户输入、选择或确认一个事件。
- 使用字符串、浮点数和 URL。
- 查询系统特性（如是否支持浮点数）。
- 生成随机数。
- 以不同的方式结束脚本。

当调用脚本库的一个函数时，要用脚本库的名称，后面加一个“.”、函数名称以及一些可能需要的参数。例如，要显示一个警告框，进行如下操作：

程序清单 14-34

```
Dialogs.alert ("You have mail!");
```

函数中使用的数据类型有 Boolean、Float、Integer、String 和 Invalid。当一个数据类型既可以是 Integer，也可以是 Float 时，称这个数据类型为 Number。如果数据类型可以为任何类型，称其为 Any。所有的函数都具有相似的错误处理方法：如果参数无效（例如数据类型不对），那么函数会返回 Invalid。参数通过数值进行传递。

下面我们进一步详细地讨论脚本库。

14.7.1 决定浏览器的状态——WMLBrowser 脚本库

WMLBrowser 脚本库可以用来决定浏览器的状态。浏览器的当前状态包括当前变量的列表、当前的 WML 卡片和历史列表。

1. 处理变量

函数 setVar(name,value) 的作用是将变量 name 绑定到 value 上。函数 getVar(name) 获取变量 name 的值。如果浏览器没有和 name 匹配的变量，将返回一个空字符串。函数 refresh() 刷新显示。所以，举个例子，如果你执行了影响当前显示内容的一系列 setVar() 函数，用户将会看到这些变化。

下面的示例建立了闪动的 Wrox 徽标。使用元素 <timer> 设定一个有规律的时间间隔来激活这个函数：

程序清单 14-35

```
extern function repaint()
{
    var text = WMLBrowser.getVar("text");
    if (text == "Wrox")
        WMLBrowser.setVar("text", "");
}
```

```
else
    WMLBrowser.setVar("text", "Wrox");
WMLBrowser.refresh();
}
```

这个函数调用的WML卡片为：

程序清单 14-36

```
<card id="card1" title="Intro" onenterforward="repaint.wmls#repaint()"
    ontimer="repaint.wmls#repaint()">
    <timer value="10" />
    <p align="center">
        <br/>
        <big>$text</big>
    </p>
</card>
```

这项应用的状态保存在浏览器的变量里。脚本通过改变变量对浏览器进行操作。上例也可以扩展，添加更多的文本、简单的ASCII图形或动画。如果使用一张以上的卡片和采用真正的WBMP或GIF图像，可能会得到一个对你的WAP服务的精彩“多媒体”介绍——即使是在一部移动电话上。

WML文档里变量的寿命由调用函数 `newcontext()` 来决定，函数 `newcontext()` 可以清空所有的变量。如果你想在用户开始调用一系列卡片之前确认你已经初始化了所有的变量，这个函数就会非常有用。

2. 浏览卡片和使用历史记录

要使浏览器访问不同的站点可以使用 `go(url)` 函数，`go(url)` 可以是目前文档的一个相对 URL，而要访问历史记录中的前一个站点可以使用 `prev()` 函数。连续调用 `go()` 和 `prev()` 函数可以覆盖前面的调用过程。如果想知道浏览器当前显示卡片的 URL，可以使用 `getCurrentCard()` 函数。如果这个 URL 是一个相对于当前脚本中 URL 的地址，则返回值是一个相对 URL，如果 URL 来自于一个完全不同的站点，则返回值是一个绝对的 URL。

14.7.2 Dialogs库

使用脚本函数中的 Dialogs 库可以激活一个对话框，请求用户输入信息或确认事件。实际的对话框在不同的电话上显示的结果可能不一样。一些浏览器可能会使用一张小 WML 卡片表示对话，另外一些可能会有单独的对话框。

你可以使用一个简单的函数 `alert(text)` 来警告用户注意某个事件，如计算结果、错误或输入的值无效。或者如果想要用户在两个不同数值之间选择，可以使用 `confirm(text,choiceOK,choiceNOK)` 函数。问题的内容就是 `text` 的内容，选项是 `choiceOK` 和 `choiceNOK`。返回值是一个 Boolean。

你甚至可以通过函数 `prompt(text,defaultValue)` 让用户向对话框中输入数值。函数将会返回输入值，缺省值为 `defaultValue`。例如，下面的函数询问用户有关作者姓名的信息：

程序清单 14-37

```
extern function setname()
{
    var result = Dialogs.prompt("Who is the author?");
```

```
WMLBrowser.setVar("name", result);
WMLBrowser.go("#author");
}
```

用户的回答在浏览器里用变量 name 设定。我们可以用两个简单的 WML 卡片调用脚本程序并显示结果：

程序清单 14-38

```
<card id="intro" onenterforward="check.wmls#setname()" />
<card id="author"><p>$(name)</p></card>
```

14.7.3 特定的语言函数

Lang 库是一些有用的函数的集合，这些函数用于转换数据类型、得到系统属性以及生成随机数。

一些 WAP 浏览器不能支持浮点数。你可以调用 float() 函数查看运行脚本的设备是否支持浮点数（大部分浏览器有这个函数）。返回值是一个 Boolean 数值，true 表示这个浏览器支持浮点数。你可能还想知道设备上 integer 数值的最大值和最小值，maxInt() 返回最大值，minInt() 返回最小值。要检查某个特定值是否是一个整数或可否转换成整数，可以调用 isInt(number) 函数，如果返回值为 true，就可以调用 parseInt(string) 函数转换数值为对应的整数值。例如：

程序清单 14-39

```
var y = isInt("13.13"); //Returns true
var y = parseInt("13.13"); //Returns "13"
```

函数 isFloat(value) 和 parseFloat(string) 的作用和上述函数一样，只不过是适用于浮点数。

最后，可以调用 characterSet() 函数来得到设备支持的字符集，返回值是由 IANA(Internet Naming Authority) 设定的代表字符集的 MIBEnum 值，ISO-8859-1 的值是 4，Shift-JIS 的值是 17。

函数 abs(number)、max(number1,number2) 和 min(number1,number2) 的作用显而易见，不必赘述。

你也可以使用 random(value) 函数生成一个正的随机数，这个随机数大于等于 0，但小于等于 value 的值。要想提高随机性，可以调用 seed(value) 函数，设定一个合适的随机种子值。

Lang 库中的最后一个函数是 abort(string) 和 exit(value) 函数。这两个函数将按你指定的方式结束脚本程序。abort(string) 函数遇错误结束，exit(value) 函数正常结束。但大部分浏览器忽略这些函数中的参数。

14.7.4 浮点数

在支持浮点数的设备上支持 Float 库。Float 库提供如下的函数：

- int(number) 函数返回数字的整数部分。
- floor(number) 函数返回不大于给定数值的最大整数。
- ceil(number) 函数返回不小于给定数值的最小整数。

- round(number) 函数返回最接近给定数值的整数。
- sqrt(number) 函数取给定数值的平方根。结果的准确度取决于设备的性能。
- pow(number,number) 函数计算以第二个数为幂的第一个数字的乘方。如果第一个数字是一个负数，则第二个参数必须是一个整数。
- maxFloat() 返回有效浮点数的最大值。
- minFloat() 返回浮点数的最小值。

在下面的例子中脚本调用上述函数并用 WML 卡片中的一个简单表格显示结果：

程序清单 14-40

```
extern function Floatcheck()
{
    var a = Float.int("13.5");
    var b = Float.floor("13.5");
    var c = Float.ceil("13.5");
    var d = Float.round("13.5");
    var e = Float.sqrt("2,2");
    var f = Float.minFloat();
    var g = Float.maxFloat();

    WMLBrowser.setVar("a",a);
    WMLBrowser.setVar("b",b);
    WMLBrowser.setVar("c",c);
    WMLBrowser.setVar("d",d);
    WMLBrowser.setVar("e",f);
    WMLBrowser.setVar("g",g);
}
```

可以用下面的 WML 卡片调用脚本：

程序清单 14-41

```
<card id="card1" title="FloatCheck" >
  <onevent type="onenterforward">
    <go href="floatcheck.wmls#Floatcheck()" />
  </onevent>

  <p>
    <table columns="2" align="LR" >
      <tr><td>Float.int ("13.5")</td><td>=> $a</td></tr>
      <tr><td>Float.floor ("13.5")</td><td>=> $b</td></tr>
      <tr><td>Float.ceil ("13.5")</td><td>=> $c</td></tr>
      <tr><td>Float.round ("13.5")</td><td>=> $d</td></tr>
      <tr><td>Float.sqrt ("2,2")</td><td>=> $e</td></tr>
      <tr><td>Float.minFloat ()</td><td>=> $f</td></tr>
      <tr><td>Float.maxFloat ()</td><td>=> $g</td></tr>
    </table>
  </p>
</card>
```

14.7.5 字符串

String 库提供文本字符串操作。

1. 基本操作

调用 `length(string)` 可以返回字符串长度。如果字符串为空, `isEmpty(string)` 返回 `True`, 否则返回 `False`。你也可以调用 `charAt(string,index)` 函数来得到字符串中的单个字符。如果你给出的 `index` 值超过范围, 则会返回一个空字符串。要想得到一个以上字符, 可以使用 `substring(string,startIndex,length)` 函数, 这个函数返回一个开始于 `startIndex`、长度为 `length` 的子字符串。

2. 查找和替换

有时你需要查找一个特定的字符串。函数 `find(string,subString)` 返回 `string` 中与 `subString` 相匹配的第一个字符的序号。如果没有相匹配的字符串则返回 `-1`。你也可以用函数 `replace(string,oldSubString,newSubString)` 替换字符串中的字符, 返回值是一个新字符串, 其中所有的 `oldSubString` 都被 `newSubString` 替换了。

3. 元素

我们在这里说的“元素”不是 XML 元素——这个元素是被一个预定义的分隔字符串分开的文本。`elementAt(string,index,sepString)` 函数返回位于 `index` 位置的元素。分隔字符串在 `sepString` 中定义。第一个元素的标识为 0。要知道一个字符串中元素的数目可以调用 `elements(string,sepString)` 函数。要在一串元素中插入一个元素, 调用 `insertAt(string, elemString,number,sepString)` 函数, 它将返回一个包含新元素的新字符串。此外你也可以用另一个元素替换在某个位置的元素, 这个过程可以通过调用 `replaceAt(string,elemString,number,sepString)` 函数来实现。最后, 你可以调用 `removeAt(string,number,sepString)` 函数实现元素的移动。

14.7.6 URL

URL 库用来检查一个 URL 是否有效, 请求某个 URL 中的特定部分, 从一个 URL 上访问请求内容以及在 URL 中添加编码字符和去掉编码字符。

1. 解析

一个 URL 的格式如下:

程序清单 14-42

```
<scheme>://<host>:<port>/<path>;<params>?<query>#<fragment>
```

下面是一个有效的 URL:

程序清单 14-43

```
http://www.wrox.com:8080/wap/index.wml;3;2?author=peter#name
```

可以用 `isValid(url)` 函数检验一个字符串是否是一个有效 URL。

你可以利用一些函数得到 URL 的不同部分:

- `getScheme(url)` 函数返回访问服务器所用的协议, 在上例中, 所用的协议是 `http`。如果没有指定的协议, 则返回一个空字符串。

- `getHost(url)`函数返回服务器的主机名称，在上例中，主机名称是 `www.wrox.com`。如果没有指定的主机名称，则返回一个空字符串。
- `getPort(url)`函数返回服务器的端口号，在上例中，端口号是“8080”。如果没有指定端口号，则返回一个空字符串。
- `getPath(url)`函数返回文件或目录的路径，在上例中，路径为 `/wap/index.wml`。如果没有指定路径，则返回一个空字符串。
- `getParameters(url)`函数返回参数（很少使用），在上例中，参数为“3;2”。如果没有指定参数，则返回一个空字符串。
- `getQuery(url)`函数返回查询部分，在上例中，查询部分是“author = peter”。如果没有指定的查询，则返回空字符串。
- `getFragment(url)`函数返回片段部分，在上例中，返回 `name`。如果没有指定的片段，则返回一个空字符串。

脚本程序有一个可以用 `getBase()` 函数得到的基础URL。例如，一个位于 `http://www.wrox.com/script.wmls` 的脚本程序将这个URL作为它的基础URL。当从一个WML文档激活一个脚本时，这个WML文档叫做参考文档，参考文档的URL可以用 `getReferer()` 函数得到。

有两种类型的URL：绝对URL和相对URL。绝对URL是一个完整的URL，如 `http://www.wrox.com/index.wml`。相对URL是部分URL，如 `/index.wml`。相对URL所关联的URL叫做基础URL。可以用函数 `resolve(baseURL,embeddedURL)` 把相对URL和对应的基础URL合并成一个绝对URL。例如：

程序清单 14-44

```
var absoluteURL = URL.resolve("http://www.wrox.com","index.wml");
```

上例将会返回一个绝对URL：`http://www.wrox.com/index.wml`。下面的例子提示用户输入一个URL（缺省状态时用脚本的基础URL），解析URL，并设定WML浏览器中的变量：

程序清单 14-45

```
extern function parseUrl()
{
    WMLBrowser.newContext();

    var base = URL.getBase();

    var aUrl = Dialogs.prompt("Enter URL:", base);

    if (URL.isValid(aUrl))
    {
        var scheme = URL.getScheme(aUrl);
        var host   = URL.getHost(aUrl);
        var port   = URL.getPort(aUrl);
        var file    = URL.getPath(aUrl);
        var para    = URL.getParameters(aUrl);
        var query   = URL.getQuery(aUrl);
        var frag    = URL.getFragment(aUrl);
```

```

WMLBrowser.setVar("scheme", scheme);
WMLBrowser.setVar("host", host);
WMLBrowser.setVar("port", port);
WMLBrowser.setVar("path", file);
WMLBrowser.setVar("parameters", para);
WMLBrowser.setVar("query", query);
WMLBrowser.setVar("frag", frag);
}
else
    Dialogs.alert("Invalid URL");

WMLBrowser.refresh();
}

```

激活上述脚本的WML文档如下：

程序清单 14-46

```

<card id="show" onenterforward="parseurl.wmls#parseUrl" >
  <p>
    Scheme: $scheme <br/>
    Host: $host <br/>
    Port: $port <br/>
    Path: $path <br/>
    Parameters: $parameters<br/>
    Query: $query <br/>
    Fragment: $frag<br/>
  </p>
</card>

```

2. 向服务器请求数据

使用loadingString(url,contentType)可以从URL中直接将内容赋与一个字符串变量，但只支持文本内容（用“text/”开头的媒体类型）。在下面的例子中脚本下载了一个用分号隔开数值的数据文件：

程序清单 14-47

```

extern function getData()
{
    WMLBrowser.newContext();

    var absoluteUrl = URL.resolve(URL.getBase(), "data.txt");
    var data = URL.loadString(absoluteUrl, "text/plain");

    var title = String.elementAt(data,0,";");
    var fname = String.elementAt(data,1,";");
    var lname = String.elementAt(data,2,";");

    WMLBrowser.setVar("title", title);
    WMLBrowser.setVar("fname", fname);
    WMLBrowser.setVar("lname", lname);
}

```

```
WMLBrowser.refresh();  
}
```

数据文件 data.txt 包含下列内容：

```
WAP;Peter;Stark
```

数据以字符串形式返回。但如果发生错误，返回的 HTTP 错误代码是一个整数。

3. 添加编码字符和去掉编码字符

最后，用两个函数来实现向 URL 上添加编码字符和去掉编码字符。虽然一个 URL 中可以包括任何 ISO Latin-1 字符，但只有可打印的 ASCII 字符（ISO Latin-1 字符集的下半部分）才能用于 URL。表示非 ASCII 字符要使用一个特殊的字符码。并且 URL 中禁用一部分 ASCII 字符，这些字符只能在编码窗体中使用。幸运的是，编码极为简单：

```
%hh
```

百分号符号表示编码字符的开始，hh 是要表示的 Latin-1 字符的 16 进制编码。

escapeString(String) 函数用 URL 字符编码一个字符串，而 unescapeString(String) 函数的作用恰恰相反，它对字符串进行解码。下面的函数调用将会返回 “http % 3 a % 2 f % 2 f w w w . w r o x . c o m % 2 f i d e x . w m l % 3 f x % 3 d % 7 f ”。

```
URL.escapeString("http://www.wrox.com/index.wml?x=\u007f");
```

14.8 如何得到更多的信息

WAP 论坛的独立站点 <http://www.wapforum.org> 上有各种规范、更新的新闻以及和 WAP 相关的事件和产品信息。另一个不错的相关站点是开发者论坛——<http://www.wapdevelopers.org>，在这里你可以测试你的 WAP 服务以及得到一个 WAP 徽标。

另外一些可以查找 WAP 信息的好地方是 WAP 浏览器厂家的站点，如：

Nokia, <http://www.nokia.com/>

Ericsson, <http://www.ericsson.se/WAP/>

Phone.com Inc., <http://www.phone.com/>

上述站点除了向开发者提供工具、规范和指导外，还提供 WAP 开发工具。众所周知，在桌面系统中 Internet Explorer 和 Netscape Navigator 之间存在差异，同样 WAP 浏览器之间也不相同。因此，一套开发应用工具并不能保证你的 WML 文档在所有的电话上都有很好的效果。

14.9 小结

在本章中我们介绍了一种 XML 应用——WAP，它将一系列移动技术带到 Web 中。我们讨论了各种产品的差异（如屏幕尺寸、可用的带宽以及处理能力等）给页面开发方式带来的影响。

我们讨论了 WML——为移动设备应用程序开发设计的标记语言。正如我们所看到的，WML

很像是 HTML 的精简版。我们也简单讨论了 WAP 规范中提供的脚本语言——WMLScript，WMLScript 和 ECMScript（基于 JavaScript）非常相似。

如果想用 XML 中的内容建立页面，我们可以将 XML 转换成许多其他语言，如 HTML 的各种版本和其他的 XML 应用。所以，我们有可能将 XML 转换为适应各个不同用户需求的应用程序，而无须为各个站点写不同版本的应用程序。

希望这些内容能引起大家的兴趣，从而去开发新一代与 Web 连接更紧密的移动产品。