

目 录

第 1 章 认识 WAP	(1)
1.1 WAP 论坛.....	(2)
1.2 WAP 的组成及主要特点	(3)
1.3 WAP 应用	(5)
1.3.1 WAP 应用优势	(6)
1.3.2 WAP 应用限制	(7)
1.3.3 WAP 应用现状	(8)
1.3.4 WAP 未来	(10)
1.4 著名的 WAP 解决方案	(11)
1.4.1 Nokia WAP 解决方案	(11)
1.4.2 Motorola WAP 解决方案	(12)
1.4.3 Ericsson WAP 解决方案	(12)
1.4.4 常见 WAP 手机	(13)
1.5 手机蓝牙技术简介	(15)
本章小结	(17)
第 2 章 WAP 原理、架构与开发工具包	(19)
2.1 WAP 协议层组成及内容	(19)
2.2 WAP 工作原理及系统架构	(21)
2.2.1 WAP 工作原理	(22)
2.2.2 Internet 与 WAP 的系统架构	(23)
2.2.3 WAP 与 Internet 的比较	(26)
2.2.4 WAP 网络服务方案	(27)
2.3 WAP 测试环境	(28)
2.3.1 浏览器环境	(28)
2.3.2 模拟环境	(29)
2.3.3 实际环境	(30)
2.4 WAP 开发工具包	(30)
2.4.1 Ericsson WapIDE	(30)
2.4.2 Nokia WAP Toolkit	(32)
2.4.3 Phone.com UP.SDK	(33)
本章小结	(35)

第 3 章 WAP 手机上网设置	(36)
3.1 WAP 手机上网前的准备工作	(36)
3.2 WAP 手机上网设置举例	(36)
3.2.1 诺基亚 7110 上网设置	(37)
3.2.2 爱立信 R320sc 上网设置	(37)
3.2.3 摩托罗拉 WAP 手机上网设置	(38)
3.2.4 西门子 3568i 手机上网设置	(40)
本章小结	(41)
第 4 章 WAP 网站的服务器建设	(42)
4.1 Web 服务器构建概述	(42)
4.2 IIS 的安装基本配置	(44)
4.2.1 IIS 响应客户请求的方法	(44)
4.2.2 IIS 的支持服务	(46)
4.2.3 IIS 的服务账号	(48)
4.2.4 IIS 的安装	(49)
4.2.5 IIS 的基本配置	(51)
4.3 WWW 服务器的建设管理	(53)
4.3.1 创建新的 Web 站点	(53)
4.3.2 配置 Web 站点	(55)
4.3.3 配置主目录	(58)
4.3.4 配置虚拟目录	(62)
4.3.5 设置主页文件	(63)
4.3.6 目录安全设置	(64)
4.3.7 Web 站点负载的多台 IIS 服务器分布	(66)
4.3.8 单站点服务器配置多个 Web 站点	(67)
4.4 建立和配置 WAP 站点服务器	(69)
4.5 PWS 的安装设置与 WAP 服务器配置	(71)
4.5.1 PWS 的安装	(72)
4.5.2 PWS 的设置	(73)
4.5.3 设置建立 WAP 服务器	(76)
4.6 建立免费个人 WAP 网站的简易方法	(79)
本章小结	(79)
第 5 章 WML 语言基础	(81)
5.1 WML 的简单例子及编辑、测试方法	(81)
5.1.1 WML 与 WAP 设备	(81)
5.1.2 使用文本编辑器编写 WML 程序	(82)
5.1.3 使用微浏览器测试 WML 程序	(83)

5.1.4	使用 WAP 开发工具包编辑和测试 WML 程序.....	(84)
5.2	WML 程序结构.....	(85)
5.2.1	WML 的元素和标签.....	(85)
5.2.2	WML 程序结构形式及组成的实例分析	(86)
5.2.3	WML 程序的基本结构.....	(89)
5.3	WML 语言的基本知识.....	(90)
5.3.1	WML 的字符集及编码.....	(90)
5.3.2	WML 字符使用基本规则.....	(92)
5.3.3	变量	(94)
5.3.4	WML 核心数据类型.....	(95)
5.3.5	WML 数据值性质.....	(96)
5.3.6	卡片与卡片组	(98)
5.3.7	卡片组模板	(100)
5.3.8	WML 与 URL、程序段锚点.....	(100)
5.3.9	浏览器操作历史	(102)
	本章小结	(102)
	第 6 章 WML 编程	(103)
6.1	卡片、卡片组及其元素.....	(103)
6.1.1	共有属性	(103)
6.1.2	WML 程序的文件头.....	(104)
6.1.3	wml 元素	(104)
6.1.4	template 元素.....	(105)
6.1.5	card 元素	(107)
6.1.6	head 元素.....	(109)
6.1.7	access 元素	(109)
6.1.8	meta 元素.....	(110)
6.2	任务及其元素	(112)
6.2.1	go 任务	(112)
6.2.2	prev 任务	(114)
6.2.3	refresh 任务	(115)
6.2.4	noop 任务	(116)
6.3	事件及其元素	(116)
6.3.1	do 元素	(117)
6.3.2	ontimer 事件	(119)
6.3.3	onenterforward 事件.....	(120)
6.3.4	onenterbackward 事件.....	(122)
6.3.5	onpick 事件	(124)

6.3.6	onevent 元素	(124)
6.3.7	postfield 元素	(126)
6.3.8	卡片与卡片组的任务替代	(127)
6.4	变量设置元素与变量设置的有关规定	(131)
6.4.1	setvar 元素	(131)
6.4.2	变量设置	(132)
6.4.3	变量定义和设置举例	(133)
6.5	用户输入处理元素	(136)
6.5.1	input 元素	(136)
6.5.2	select 元素	(139)
6.5.3	option 元素	(142)
6.5.4	optgroup 元素	(143)
6.5.5	fieldset 元素	(145)
6.6	锚、图像、定时器及其元素	(146)
6.6.1	anchor 元素	(146)
6.6.2	a 元素	(147)
6.6.3	img 元素	(148)
6.6.4	timer 元素	(150)
6.7	文本格式化及其元素	(152)
6.7.1	增强元素	(152)
6.7.2	br 元素	(153)
6.7.3	p 元素	(154)
6.7.4	td 元素	(155)
6.7.5	tr 元素	(155)
6.7.6	table 元素	(156)
本章小结		(158)
第 7 章 WMLScript 语法基础		(163)
7.1	简单例子: WML 程序中调用 WMLScript 函数	(163)
7.2	WMLScript 的主要优点及其字节码解释器	(164)
7.2.1	使用 WMLScript 的主要优点	(165)
7.2.2	WMLScript 的字节码解释器	(165)
7.3	WMLScript 基本规则	(167)
7.3.1	WMLScript 与 URL	(167)
7.3.2	词法结构	(168)
7.3.3	WMLScript 程序的基本书写规则	(171)
7.4	变量与数据类型	(172)
7.4.1	变量及其声明	(172)

7.4.2	变量的作用域与生命期.....	(174)
7.4.3	变量的使用	(175)
7.4.4	变量类型与数据类型	(175)
7.4.5	变量值域	(176)
7.5	操作符与表达式	(177)
7.5.1	赋值操作符	(178)
7.5.2	数学运算操作符	(179)
7.5.3	位操作符	(181)
7.5.4	逻辑操作符	(182)
7.5.5	比较操作符	(183)
7.5.6	其他几种操作符	(185)
7.5.7	表达式	(187)
7.6	数据类型自动转换规则.....	(187)
7.6.1	一般转换规则	(187)
7.6.2	操作符数据类型转换规则.....	(189)
7.6.3	操作符与数据类型汇总.....	(190)
	本章小结	(192)
	第 8 章 WMLScript 脚本程序设计.....	(193)
8.1	语句	(193)
8.1.1	基本语句	(194)
8.1.2	条件语句	(197)
8.1.3	循环语句	(198)
8.2	函数的声明与调用	(203)
8.2.1	函数的声明	(203)
8.2.2	函数的调用	(205)
8.2.3	函数的嵌套调用	(207)
8.3	预编译	(208)
8.3.1	外部文件	(209)
8.3.2	访问权限	(210)
8.3.3	Meta 信息	(211)
8.4	运行错误检测和处理	(212)
8.4.1	错误检测	(212)
8.4.2	错误处理的两类情况	(212)
8.4.3	致命错误及其处理	(213)
8.4.4	非致命错误及其处理	(215)
	本章小结	(218)

第 9 章 WMLScript 库及库函数	(219)
9.1 Lang 库及其函数	(219)
9.1.1 abs 函数	(219)
9.1.2 min 函数	(220)
9.1.3 max 函数	(221)
9.1.4 parseInt 函数	(221)
9.1.5 parseFloat 函数	(222)
9.1.6 isInt 函数	(222)
9.1.7 isFloat 函数	(223)
9.1.8 maxInt 函数	(223)
9.1.9 minInt 函数	(224)
9.1.10 float 函数	(224)
9.1.11 exit 函数	(224)
9.1.12 abort 函数	(225)
9.1.13 random 函数	(225)
9.1.14 seed 函数	(226)
9.1.15 characterSet 函数	(226)
9.2 Float 库及其函数	(227)
9.2.1 int 函数	(227)
9.2.2 floor 函数	(227)
9.2.3 ceil 函数	(228)
9.2.4 pow 函数	(228)
9.2.5 round 函数	(229)
9.2.6 sqrt 函数	(229)
9.2.7 maxFloat 函数	(230)
9.2.8 minFloat 函数	(230)
9.3 String 库及其函数	(230)
9.3.1 length 函数	(231)
9.3.2 isEmpty 函数	(232)
9.3.3 charAt 函数	(232)
9.3.4 subString 函数	(233)
9.3.5 find 函数	(234)
9.3.6 replace 函数	(235)
9.3.7 elements 函数	(235)
9.3.8 elementAt 函数	(236)
9.3.9 removeAt 函数	(237)
9.3.10 replaceAt 函数	(237)

9.3.11	insertAt 函数.....	(238)
9.3.12	squeeze 函数.....	(239)
9.3.13	trim 函数.....	(239)
9.3.14	compare 函数.....	(240)
9.3.15	toString 函数.....	(241)
9.3.16	format 函数.....	(241)
9.4	URL 库及其函数	(243)
9.4.1	isValid 函数	(243)
9.4.2	getScheme 函数.....	(244)
9.4.3	getHost 函数.....	(244)
9.4.4	getPort 函数.....	(245)
9.4.5	getPath 函数	(245)
9.4.6	getParameters 函数.....	(246)
9.4.7	getQuery 函数	(246)
9.4.8	getFragment 函数	(247)
9.4.9	getBase 函数.....	(247)
9.4.10	getReferer 函数.....	(247)
9.4.11	resolve 函数.....	(248)
9.4.12	escapeString 函数	(248)
9.4.13	unescapeString 函数	(249)
9.4.14	loadString 函数.....	(249)
9.5	WMLBrowser 库及其函数	(250)
9.5.1	getVar 函数.....	(250)
9.5.2	setVar 函数	(251)
9.5.3	go 函数	(251)
9.5.4	prev 函数.....	(252)
9.5.5	newContext 函数	(252)
9.5.6	getCurrentCard 函数.....	(253)
9.5.7	refresh 函数	(253)
9.6	Dialogs 库及其函数	(254)
9.6.1	prompt 函数.....	(254)
9.6.2	confirm 函数.....	(254)
9.6.3	alert 函数	(255)
9.7	WMLScript 非标准库及其库函数	(256)
9.7.1	openFile 函数	(256)
9.7.2	closeFile 函数.....	(257)
9.7.3	println 函数.....	(257)

9.8 WML/WMLScript 应用举例	(258)
9.8.1 WMLScript 库函数应用举例	(258)
9.8.2 数值范围有效性检验实例	(259)
9.8.3 货币换算实例	(262)
9.8.4 简单动画实例	(268)
本章小结	(270)
第 10 章 HDML 编程	(271)
10.1 HTML 语言基础知识	(271)
10.1.1 HTML 页面	(271)
10.1.2 HTML 页面文件的结构	(272)
10.1.3 HTML 页面编程简例	(274)
10.2 HTML 标签及使用	(275)
10.2.1 文本类标签及其属性	(275)
10.2.2 图像标签及其属性	(278)
10.2.3 列表类标签及其属性	(280)
10.2.4 表格类标签及其属性	(281)
10.2.5 文档超链接标签	(284)
10.2.6 表单类标签与交互界面	(286)
10.2.7 框架类标签及其属性	(292)
10.3 HDML 语言编程基础	(294)
10.3.1 HDML 语言的开发环境	(295)
10.3.2 HDML 页面	(295)
10.3.3 HDML 页面文件的结构	(296)
10.4 HDML 标签及编程	(297)
10.4.1 文本标签及规定	(297)
10.4.2 超链接标签	(299)
10.4.3 图像显示标签	(300)
10.4.4 选单标签	(301)
10.4.5 行为(ACTIVITY)及其嵌套	(302)
10.4.6 变量的定义与引用	(303)
10.4.7 获取用户输入	(304)
10.4.8 ACTIVITY 间的参数传递	(305)
10.4.9 TASK 属性的取值	(306)
10.4.10 HDML 的 CGI 编程	(306)
本章小结	(307)
第 11 章 WAP 编程与开发的高级技术	(308)
11.1 汉字与图像的使用问题	(308)

11.1.1	汉字使用与字符集转换.....	(308)
11.1.2	图像使用与图像格式转换.....	(310)
11.2	ASP 和数据库技术在 WAP 开发中的应用	(310)
11.2.1	在 WML 程序中使用 ASP.....	(310)
11.2.2	在 ASP 中编写 WML 程序.....	(312)
11.2.3	利用 ASP 在 WML 中实现动态数据库应用.....	(315)
11.3	PHP 编程在 WAP 开发中的应用	(315)
11.3.1	基本规则	(316)
11.3.2	程序举例	(317)
11.4	Perl 与 WAP 的综合应用.....	(318)
11.4.1	基本规则	(318)
11.4.2	程序举例	(320)
11.5	C/C++ 与 WAP 的综合应用	(325)
11.5.1	基本规则	(326)
11.5.2	程序举例	(326)
11.6	JSP 技术在 WAP 开发中的应用	(328)
11.6.1	基本规则	(329)
11.6.2	程序举例	(331)
11.7	Servlet 技术在 WAP 开发中的应用	(334)
11.8	HTML 过滤器和 HTML 页面的转换	(336)
11.8.1	Wapitout.....	(336)
11.8.2	TransWap	(337)
11.8.3	Coolie.....	(338)
	本章小结	(339)
	第 12 章 WAP 安全与实现.....	(340)
12.1	数据加密原理与实现方法.....	(340)
12.1.1	基本概念	(341)
12.1.2	基于单钥技术的传统加密方法.....	(342)
12.1.3	改进的传统加密方法	(343)
12.1.4	基于双钥技术的现代加密方法.....	(346)
12.2	实现 WAP 安全的一般方法	(353)
12.2.1	数字签名	(353)
12.2.2	数字时间戳	(354)
12.2.3	数字凭证	(355)
12.2.4	认证中心	(357)
12.3	WAP 数字凭证的使用与防范	(359)
12.3.1	WAP 客户端凭证的使用	(360)

12.3.2	WAP 服务器/网关凭证的使用	(361)
12.3.3	数字凭证认证操作的安全防范及问题.....	(364)
12.4	WAP 服务器凭证的申请操作与 WAP 服务器配置	(365)
12.4.1	WAP 服务器凭证的申请操作	(365)
12.4.2	Nokia WAP 服务器中的凭证安装.....	(370)
	本章小结	(372)
	参考文献	(373)

第 1 章 认识 WAP

无线应用协议 WAP(Wireless Application Protocol)也称为无线应用程序协议,是在数字移动电话、Internet 及其他个人数字助理机 PDA、计算机应用之间进行通信的开放性全球标准。WAP 由一系列协议组成,用于标准化无线电通信设备,也可用于 Internet 访问,包括收发 E-mail、访问 WAP 网站上的页面等等。WAP 将移动网络和 Internet 以及企业的局域网紧密地联系起来,提供了一种与网络类型、运行商和终端设备都独立的、无地域限制的移动增值业务。通过这种技术,无论用户身在何地、何时,只要通过 WAP 手机,即可享受无穷无尽的网上信息资源。

利用 WAP 实现的网络业务以其移动性、灵活性、个人化、信息实时性、信息简短实用而受到广大数字移动电话用户的普遍欢迎。据预测,到 2002 年,中国 Internet 用户将达 6500 万,其中移动上网的用户可达 3000 万,超过使用 PC 上网的用户;到 2004 年,全球 Internet 用户将突破 10 亿,其中约有 3.5 亿用户将通过移动方式接入 Internet。显然,用于解决无线移动接入的 WAP 技术将发挥愈来愈重要的作用。与此相应,WAP 网站建设和 WAP 编程将成为一项十分热门的工作。为了帮助大家全面而深入地学习 WAP 技术,本书将详细讲解 WAP 协议的组成及工作原理,深入讲解 WML 语言的规则、编程方法和开发环境。同时,通过大量实例,详细讲述从建立 WAP 的 Web 服务器、搭建 WAP 平台,到开发 WAP 动态网页和发布 WAP 网页的各种技术。本书同时配有一张光盘,含有针对本书内容的多媒体教学课件、全书所有举例的源程序以及相关的开发工具和软件系统。

为了便于大家更好地了解 WAP 的基本情况,我们先在本章对 WAP 的发展历史、形成背景及相关情况等作一简单介绍。



1.1 WAP 论坛

近几年来, Internet 及其应用获得了迅速发展, 并带来了信息技术的革命, 同时也促进了移动电话之类的移动通信设备的巨大发展。随着 Internet 的普及, 接入 Internet 的个人计算机用户可以极其方便地获取世界各地的网络信息资源, 所以移动通信用户也对无线电通信设备的信息服务功能提出了更高的要求。移动通信网络商们也不得不想办法来扩展移动网络, 并提供更多的功能和无线电信息服务。因此, 人们想到了将移动网络与 Internet 结合起来的办法, 利用 Internet 来为移动通信用户提供高效率的信息服务。

为了实现 Internet 与移动网络的集成并抢占无线电通信的市场, 最初许多厂商纷纷推出自己的互联标准。虽然说这在一定程度上满足了移动通信用户的需求, 但不同标准之间的不兼容性所导致的弊端马上突显出来。这种形势下, 为了协调和规范市场上各种各样的技术标准, 促进无线网络与 Internet 的互联, WAP 就应运而生了。

1997 年 6 月, Unwired Planet(Phone.com)、Ericsson、Motorola 和 Nokia 四大通信公司合作, 成立了 WAP 论坛(WAP Forum)。其宗旨就是将 Internet 的海量信息与先进的业务引入到无线电通信设备使用领域中, 目标是建立一个能够协调不同无线网络技术的全球无线协议规范。该论坛成立初期只有 4 名成员, 但他们广泛邀请无线行业中的其他伙伴加入进来, 目前已经有了 500 多名成员, 拥有全球移动电话 90% 以上的份额, 并代表着超过 1 亿用户的电信公司、领先基础设备提供商、软件开发商以及向无线行业提供解决方案的相关机构等。

WAP 论坛从成立伊始就致力于 WAP 的开发和研究, 拟确立一个世界范围内适用的, 基于 Internet 并为巨大无线市场服务的标准。1997 年 9 月, 该论坛发布了 WAP 标准的构架。1998 年 1 月, 论坛发起成员建立了 WAP 论坛有限公司(WAP Forum, Ltd.)来管理 WAP 协议建立过程中的有关事务。同年 4 月, WAP 论坛发布了 WAP 1.0 版, 1999 年 6 月 30 日又发布了 WAP 1.1 版, 是年 12 月又推出了 WAP 1.2 版。

针对 WAP 开发者和程序员的注册证书在 1999 年 8 月份发布, 并开始注册服务。2000 年 4 月, WAP 论坛又发布了新的产品证书。



1.2 WAP 的组成及主要特点

总体来说，WAP 的组成及特点主要包括以下几个方面：

(1) WAP 提供了一套开放、统一的技术平台，用户使用移动设备可以很容易地访问和获取以统一的内容格式表示的 Internet 或 Intranet 信息及各种服务。比如综合新闻、天气预报、股市动态、商业报道、当前汇率和商业信息等等。随着 WAP 应用的深入，电子商务、网上银行将来也会在 WAP 上逐步实现。用户还可以通过 WAP 随时随地获得体育比赛结果、娱乐圈趣闻以及幽默故事等，为生活增添情趣；也可以利用 WAP 的网上预定功能，把生活安排得有条不紊。

(2) WAP 支持目前常用的绝大多数无线电设备，包括移动电话、FLEX 寻呼机、双向无线电通信设备等。这些设备相对于台式个人计算机而言，其 CPU 功能较弱，内存较少，无线环境下电力供应有限，显示屏较小，输入功能有限。另外，在传输网络上，WAP 支持目前的各种移动网络，如 GSM、CDMA、PHS 等，并可支持未来的第三代移动通信系统。不过相对使用 Internet 的有线网络带宽而言，无线网络的带宽资源终究是有限的。因此，考虑到以上的限制和不利因素，WAP 充分借鉴了 Internet 的思想，并进行了一定的修改和简化，采用标准的数据格式来表示应用程序和网络内容，采用与在 PC 上类似的浏览器软件作为 WAP 访问的微浏览器(MiniBrowser)，并采用标准的通信模式进行上网浏览，从而实现无线网络信息服务。

(3) WAP 还同时定义了一套软硬件的接口。通过这些接口的移动设备和网站服务器，人们可以像使用 PC 一样，使用移动电话收发 E-mail 和浏览 Internet。前文已述，WAP 是一种通信协议，它不仅提供了应用开发和运行环境，而且对当前流行的嵌入式操作系统 PalmOS、EPOC、WindowsCE、FLEXO、JavaOS 等提供了广泛的支持。

(4) WAP 标准还定义了一种应用环境 WAE(Wireless Application Environment)，能够让设计人员开发独立于设备的用户界面，并可使用 WML 脚本 WMLScript 的 WAP 编程语言，把可执行的逻辑嵌入到移动终端中。这样，移动终端上就可以运行一种微型浏览器，供无线用户浏览信息。这种微型浏览器与 PC 机上的 IE 或 Netscape 浏览器极为类似。

无线标记语言 WML(Wireless Markup Language)用来显示各种文字、图像等数据。WML 是一种基于扩展标记语言 XML(Extension Markup Language)的语言，是 XML 的子集。



而作为 WML 的脚本语言, WMLScript 可以补充 WML 的一些限制, 如实现对用户输入数据的有效性进行检查等, 这一方面增强了 WML 的浏览和表示功能, 另一方面对用户的操作也给予了更加灵活和智能的处理。

WAP 应用环境 WAE 是一种普遍意义上的应用开发框架, 对在不同的无线电通信网络上开发和运行 WAP 应用服务提供了广泛支持。目前这一框架主要基于现有的 Internet 技术。

(5) WAP 应用结构与 Internet 结构非常类似。典型的 WAP 应用系统定义了 3 类实体:

① 具有 WAP 用户代理功能的移动终端(Client)。典型的终端, 比如 WAP 手机, 实际上相当于 Internet 中的普通 PC。终端显示屏上运行有微浏览器, 用户可以采用简单的选择键来实现 WAP 服务请求, 并可以通过无线电通信方式发送和接收所需信息。当前, WAP 移动终端主要使用 WML 来显示各种文字、图像等数据。作为 XML 的子集, WML 主要用于标记和说明 WAP 移动终端收发的 Internet 信息及用户接口, 使得开发人员能够采用与设备独立的方式定义 WAP 应用的用户接口。

而且, 多数情况下, WAP 还使用 WMLScript 直接在移动终端上处理警告等消息, 避免移动终端和远端服务器之间的数据交互, 从而可以减少带宽资源的消耗。

② WAP 代理。它通过协议网关, 能够实现 WAP 协议栈, 包括 WSP、WTP、WTLS、WDP 等(这些协议的含义随后讨论)与 Internet 协议栈之间的转换。WAP 代理还提供了信息内容编解码器(Content Encoders and Decoders), 可以把 WAP 数据压缩编码, 从而减少网络数据流量, 最大限度地利用当前无线网络缓慢的数据传输速率。此外, WAP 代理还采用了错误校正技术, 可以确保网络浏览和数据传输过程中, WAP 通信不会因为无线电通信线路质量的变化而受到严重影响。

③ 源数据服务器(Origin Server)。这是 WAP 应用系统中规模最大的实体, 旨在为 WAP 应用提供数据服务支持, 比如支持 WAP 的 Web 网站以及相关的网站服务等。WAP 的 Web 服务器中通常有采用 WMLScript 编写的 WAP 应用, 这些应用不仅可以根据 WAP 移动终端的需要而被随时下载, 而且还可以在不需要的时候从 WAP 终端中全部卸除。

(6) WAP 由一系列通信协议组成。WAP 的协议栈采用了层次化设计, 从而为应用系统的开发提供了一种可伸缩和可扩展的环境。每层协议栈均定义了相应的接口, 可被上一层协议所使用, 也可被其他服务或应用程序所直接应用。设计时, WAP 充分借鉴了 Internet 的协议栈思想, 并进行修改和简化, 使之能够有效地适用于无线应用环境。WAP 的各层协议及含义说明如下:



① WTP(Wireless Transaction Protocol)。即 WAP 无线电传输协议，用于提供轻量级的面向事务处理的服务，可以专门优化并适用于移动终端的设计。

② WDP(Wireless Datagram Protocol)。即 WAP 无线电数据报协议，用于传输数据，发送和接收消息。

③ WSP(Wireless Session Protocol)。即 WAP 无线会话层协议，主要为上层的 WAP 应用提供面向连接的、基于 WTP 的会话通信服务，或基于 WDP 的无连接、可靠的通信服务。

④ WTLS(Wireless Transport Layer Security)。指 WAP 的无线传输安全层协议，是基于 SSL 的安全传输协议，主要为数据传递提供安全服务。

⑤ HTTP 接口。主要用于支持移动终端对 Internet 内容的信息检索请求。

WAP 还提供了通用的数据传输服务，可以支持多种无线承载网络，使得上层的 WAE、WSP、WTP、WTLS 能够独立于下层的无线网络，从而使全球性的网络交互操作得以实现。当然，传输的数据量和用户交互的本质会影响运营商对所用网络的选取；但不论选取哪一种无线承载网络，其目标只有一个，即达到最大的网络服务效率。

(7) 除了 WAE 和通信协议以外，WAP 标准还定义了无线电话应用 WTA(Wireless Telephony Applications)。WTA 使得 WAP 可以很好地与目前电信网络中现存的各种先进电信业务相结合，如智能网 IN(Intelligent Network)业务。通过使用浏览器方式的用户接口，移动用户可以直接应用各种智能网业务，而无需修改移动终端配置。

(8) 一般说来，WAP 移动终端主要采用 WML 和两种 WAP 服务器进行通信。这两种服务器分别是 WAP 代理服务器和 WTA 服务器。WAP 代理可以把 WAP 请求翻译成为 WWW 请求，从而移动终端就可以向 Web 服务器提交 WWW 请求；WAP 代理同时也可把 Web 服务器的响应转换成压缩的二进制 WML 格式的数据，以便能被移动终端所接受。

1.3 WAP 应用


近几年来，Internet 和移动电话这两项技术的发展已经给亿万人的生活带来了深刻影响。Internet 能使人们十分方便而廉价地获得大量信息，为此用户所需设备至少是一台调制解调器和电话线，以及通过它们与 Internet 服务提供商 ISP 相连的个人计算机。而移动电话打破了用户计算机与 ISP 有线接入的束缚，用户可以不必要坐在办公桌旁或家中的固定电话旁上网了。通过移动电话和 WAP 技术，用户可以漫游到其他地方，并且仍能继续与 Internet 接

驳，与家庭、朋友、业务同事和客户相互联系。

WAP 应用的趋势是，Internet 和移动电话两种技术将结合在一起，使信息的接入不仅不受信息源的限制，而且不再受接入者位置的限制。诺基亚(Nokia)公司率先推出了允许移动 Internet 接入的产品，这种产品就是 WAP 手机。

WAP 手机是集移动电话与移动电脑于一身的新型通讯工具，不仅具有普通手机的功能，而且还具有收发 E-mail、传真、浏览新闻、查看股市行情等功能。WAP 手机与一般手机的不同之处在于，它内置有微型浏览器(MiniBrowser)、缓存(Cache)和内存，并支持客户端的 Cookies 和 Session。类似的地方在于，普通计算机上网需要使用 IE 浏览器或 NetScape 浏览器，WAP 手机上网也要使用浏览器，只不过是微型浏览器；而且 WAP 手机上网也要进行一系列的设置。

由于技术原因，WAP 手机上网和普通的计算机上网还是有很大差别的。比如，WAP 手机内存不大、屏幕较小且无线电频带较窄，所以目前 WAP 手机所显示的网页内容主要是文字，尽管也有一些较小的图片，但仅有黑、白和灰三种颜色。业界专家指出，能够让手机和手持计算机设备成为上网工具的 WAP 技术将是 Internet 技术的下一个应用热点。不过，WAP 技术的流行与广泛实现，还需要有两个前提条件，那就是一方面 WAP 必须解决好目前还不便于操作的问题，另一方面网络运营商还需加紧 WAP 网的基础建设，使 WAP 手机有用武之地。

需要强调的是，WAP 技术应用的普及不仅仅取决于移动电话以及其他终端技术的飞速发展，而且也取决于后端信息交换技术的不断完善。目前的主流技术是利用网关服务器实现 WAP 协议与 HTTP 协议的转换，为此许多通信厂商都提出了自己的端到端(End to End)的解决方案。而且，WAP 的发展还需要得到各有关方面的支持，比如电信运营商应当提供 WAP 网关服务，Internet 内容供应商 ICP 以及各种专业的在线服务供应商均需配置 WAP 服务器以提供 WAP 服务。

1.3.1 WAP 应用优势

与普通 Internet 相比，应用 WAP 协议的无线 Internet 在终端、接入方式、带宽、稳定性以及商业运作模式之间存在巨大差异。无线 Internet 可以通过掌上服务，把每个用户个体与

Internet 紧密地结合在一起, 因此用户可以得到比传统的 Internet 更多的个性化信息服务。

从 WAP 的组成和特点来看, 它具有以下应用优势:

- (1) 适用于无线数据的传输机制;
- (2) 独立于网络标准;
- (3) 开放的标准, 并独立于各生产厂商;
- (4) 可以用作 Internet 浏览器, 支持超文本链接, 具有较强的交互能力;
- (5) 可以从服务器上直接下载应用, 从而可以快速提供新的服务。这一点也是 WAP 与嵌入式软件明显不同的地方。

1.3.2 WAP 应用限制

尽管 WAP 具有较强的应用优势, 但目前也有一些条件限制了 WAP 的应用。这主要表现在 ISP 环节、网页显示内容、无线接入及通信速度等方面。

随着无线技术、Internet 接入和内容服务的日益相互融合, 许多厂商都想尽快解决在手持设备上显示各种信息的方法。虽然说这对无线上网用户而言是个好消息, 但令人遗憾的是, 用户要享受到这样的服务, 必须经由无线 ISP 这个重要的环节, 也就是说在目前的技术条件下, 这个环节是不能越过的, 解决这个问题还有待时日。

手持设备的显示屏幕非常小, 单屏显示信息有限, 这在一定程度上限制了 WAP 技术的应用。不过, 为了使手持设备用户可以访问尽可能多的网站, 获取尽可能多的信息, 许多厂商都在想办法研究信息的表现形式。例如, 美国 Digital Paths 公司开发出了 Digital Paths 技术。利用这种技术, 服务商可以把所有信息以一种简单实用的文本形式显示在手机设备上, 这可以大大改善网页内容的可读性。

另外, Digital Paths 公司还采用一台服务器作为网站与用户之间取得联系的中介。服务器上的应用软件可以根据用户手持设备的类型, 对截获的网页内容重新进行格式处理, 以实现较好的显示效果。这种方法突破了普通手持设备 160×160 像素的分辨率和手机屏幕 5 行显示空间的限制。不过这种技术真正达到标准化并获得广泛普及, 恐怕还需要一段时间。

除了上述几点外, 无线接入问题目前一时难以解决, 也会限制 WAP 应用。例如, Palm VII 这种设备就必须通过一个采用 802.11 技术的基站进行无线电连接, 虽然说已有一些厂商推出了 802.11 基站设备, 但种类和数量都十分有限, 与市场要求的标准还有很大差距。

而且, WAP 手机在收发 E-mail、浏览网站信息等功能之外, 很难提供更复杂的通信应



用,比如在线采购和视频会议等。其关键是目前移动通信网的运行速度根本无法达到要求。只有将来移动通信网进行全面“提速”,并使 Internet 实现快速移动后,更高级的 WAP 应用功能才可成为现实。

1.3.3 WAP 应用现状

目前,使用移动电话访问网上信息的方式主要有两种:其一,由电信增值服务(Telephony Value Added Services)直接在无线网络中提供 WML 的 Web 格式的信息服务;其二,通过能收发无线信号的代理服务器(Proxy)访问 Internet。

在第二种方式中,实现访问的可行方法有多种。比如,网上的 HTML 页面信息可以通过一个过滤系统,再经由代理服务器传递给手机;或者,由电信或 ISP 设立带有过滤系统的代理服务器,然后手机通过它再来访问一般的网站;还有,网站针对手机用户直接制作 WML 格式的页面,并通过代理服务器与用户手机进行交互。

当前,通过 WAP 协议上网的终端设备主要是移动电话和掌上电脑,然而 WAP 的应用并不仅限于此。将来,MP3 网络 CD 机、电子书籍阅读机等都会成为使用 WAP 协议的无线电 Internet 终端。而且,随着计算成本的不断降低,将来还会有更多的具有单一功能的信息产品出现,这些产品都有可能利用 WAP 协议进行无线连接。随着 WAP 应用深入,移动通信与移动计算将更加紧密地结合在一起,日益智能化和超大记忆容量的设计还将使具备上网功能的 WAP 移动电话与利用 WAP 协议上网的掌上电脑最终成为同一类终端产品。

进入 2000 年以来,无线电数据通信领域发展十分迅速且利润增加迅猛。最初,人们主要通过普通的 SMS 向移动电话发送文本消息,比如新闻、天气预报等,但 SMS 所能提供的交互性十分有限。现在大多数人已经不能满足于这种低层次的服务,为改变这一局面,全球的网络运营商、终端和基础设施的制造商和信息提供商正以前所未有的热情,通力合作,研究开发新的服务模式。Microsoft 公司宣布开发支持 WAP 电话的浏览器,即 Microsoft Mobile Explorer,这是一种双模微浏览器,可以显示符合 HTML 和 WAP 1.1 两种标准的网页信息。这种浏览器帮了运营商和开发商的大忙,他们无需再在两种技术之间进行两难选择了。Microsoft Mobile Explorer 还是一种模块化的移动电话平台,支持安全的公司数据访问、E-mail 收发、Internet 浏览以及基于位置的信息业务,甚至从 WAP 电话和掌上终端进行的电子商务。

世界许多国家对于 WAP 的应用是非常积极的。以诺基亚的故乡芬兰为例来说,移动通信的发展已经给人们的生活带来了实质性的便利和享受。芬兰的手机普及率达 60% 以上,为全世界之首(2000 年初)。除了交谈之外,人们已经开始利用手机传输大量数据和大量短信息,或利用手机来了解天气预报、银行账户赢余、交通信息甚至美国 CNN 的头条新闻等。夫妻之间常常通过手机留言,告诉对方什么时候回家、什么时候接送孩子或顺路到超市买什么食品。越来越多的芬兰青年在搬入新居的时候不安装固定电话,许多公司的办公室也开始使用移动电话。

除了日常生活中的应用之外,在办公环境领域,芬兰许多企业的现有局域网逐步延伸为更灵活的无线电局域网。比如芬兰首都赫尔辛基近郊的一家名为 Anygraaf 的图像设计公司,于 1999 年 4 月撤走了办公桌上所有的普通有线电话,代之以一套无线电数据网络和 10 多部诺基亚 6150 手机。公司员工之间相互联系时只需拨 3 位数分机号码,即可建立连接,而不论对方是近在咫尺,还是远在其他国家。随后,该公司又把这套移动通讯系统同公司的终端相联,这样,员工就可以利用自己的手机收发 E-mail 及公司资料了。

无线网络的竞争给欧洲大多数国家提供了启动以技术驱动的新经济的最好机会。GSM 系统在欧洲大陆得到了统一的应用,现在,该系统已经广泛应用于包括中国在内的 120 多个国家。由于欧洲大陆正在迅速成为世界领先的 Internet 服务的最大实验室,所以无论是外地还是本地的公司都向欧洲注入了大量的研究资金。朗讯公司、德州仪器公司、日本电气公司、加拿大北方电讯公司等纷纷加强他们在欧洲的移动电话业务。

瑞典 Melody 公司的一个小组为北欧公司开发电话网页,帮助当地的商店如街角的快餐店等通过发送信息给过路人的手机以拉拢顾客。总部位于斯德哥尔摩的 Nocom 专门制作用于发送到手机上的微型公司网址,该公司股票于 1999 年 1 月在斯德哥尔摩交易所上市,当年底股价已从最初的每股 5.35 美元飚升至 16.25 美元。法国 Webraska 公司专门出售巴黎交通堵塞的最新图像,堵塞情况司机们可以从手机的显示屏上浏览。

许多美国和日本公司,如 AT&T、Sony 等,为了防止竞争对手抢先而纷纷在移动网络上掷下重金。1999 年 10 月 5 日, MCI World.com 公司斥资 1150 亿美金收购斯普林特公司,其主要原因就是看中了后者强大的移动网络,以及把其 Internet 延伸到无线领域的美好前景。与此同时,传媒巨子如有线新闻电视网、路透社和 ESPN 等都在推销新闻板块,让广大用户可以通过手机浏览他们的新闻。

再看 WAP 在中国的应用情况。1999 年 11 月,美通公司宣布开通了第一个 Internet



门户网站, 即掌门网(www.byair.com), 专门支持 WAP 手机、掌上电脑等掌上终端的无线 Internet 接入, 并提供比较丰富的信息资源及个人电子商务服务。招商银行在 2000 年 2 月初率先推出基于 WAP 的“移动银行”服务, 让客户能通过移动电话与银行业务系统连接, 在移动电话界面上直接完成各种银行业务的服务, 如账户查询、自动缴费、多功能转账和证券服务, 以及股市信息查询等。中国银行也宣布与中国移动通信公司一起, 联合开发“移动银行”服务, 并推出类似业务。中国移动通信公司于 2000 年 5 月中旬开通了全国范围内的无线 Internet 服务, 全球通用户可以通过 WAP 手机实现网上信息浏览、信息查询、信息点播、网上炒股、漫游炒股、网上电子商务等功能。同时, 新浪、网易、搜狐等一些知名的 Internet 内容服务商(ICP)也将相继推出 WAP 信息服务。

1.3.4 WAP 未来

1999 年全球有 1.35 亿移动电话用户, 根据 IDC 公布的调查报告, 到 2005 年, 全球无线电通信用户预计将达到 10 亿。调查中还发现了一个值得注意的现象, 即许多手机用户, 往往同时也是 Internet 用户, 这些人在本国内的所占比例, 通常决定了该国的通信市场大小。一般来说, 这些人所占比例越高, 他们国家的通信市场就越发达。比如日本, 由于对移动电话技术和相应的基础设施进行了巨额投资, 使得日本移动电话的用户很早便享受到了互动信息业务, 虽然一时还不完备, 但却极大地刺激了市场的发育。据预测, 到 2002 年日本的手机用户数将会达到全世界移动电话用户数的 10%。同样, 欧洲的情况也非常令人乐观。未来几年中, 欧洲移动电话市场的增长速度可能达到 300% 以上。早在 1999 年, Ellison, 这位美国 Oracle 公司的 CEO 便在瑞士日内瓦举行的“TELECOM 99”展览会上指出, Internet 将取代 PC 在历史舞台上扮演主角, 利用无线终端与 Internet 连接的用户将很快超过使用 PC 与 Internet 连接的用户。

中国信息产业部电子信息中心和蓝田市场研究公司联合进行的全国性大范围调查, 形成了国内信息产业多层面的基础数据(简称 ITD2000)。结果显示, 我国移动电话用户绝对数量超过 5000 万户, 无线寻呼用户超过 7000 万户, 已经成为名副其实的移动通信大国。据 ITD2000 显示, 10 年来, 我国移动电话的发展从无到有, 空前迅猛。移动电话用户数量一直保持高速增长。1999 年我国移动电话用户达到 4000 万户, 目前已超过 5000 万户, 仅次于美国和日本, 居世界第三位, 增长速度居世界第一。而且, 国产移动通信制造业已取得

群体性突破。在移动通信市场上，中兴、大唐、华为、金鹏等民族通信企业迅速崛起，并获得移动设备入网证。国产品牌手机占领的市场份额日益增大。

基于 WAP 的移动通信设备上网将为我们展示一个全新的空间。虽然 WAP 技术本身仍在不断地完善和发展，但随着技术的不断进步和用户市场的日益成熟，所有的问题都会逐步加以解决，与之相应，在移动通信设备中发生的这场信息革命，也将变得更加轰轰烈烈。

WAP 目前已经成为通过移动电话及其他无线移动终端使用无线信息服务的全球性的技术与工业标准，它不但使现有移动网络的许多应用得到了迅速发展，同时也催生更多的全新的增值业务，而且这些应用还将由纯信息的服务逐渐向更加交互化和电子商务化的方向发展。

如果说移动电话的出现使我们摆脱了地理位置的束缚，那么 Internet 的出现则给我们提供了一片广阔的信息天地；移动电话与 Internet 的结合，不仅使人们对于信息的获取能独立于所处的地理位置，而且还可以独立于信息的来源。所以 WAP 技术必将给我们带来一个全新的未来，人类最终将超越时间和空间对沟通的束缚，进入更为激动人心的美好时代。

1.4 著名的 WAP 解决方案

看到 WAP 技术应用的巨大市场，世界许多知名的移动通信设备公司纷纷推出自己的 WAP 解决方案，其中比较著名的方案当属三大移动设备公司 Ericsson、Motorola 和 Nokia 公司推出的解决方案。下面我们就简要介绍一下三大公司的 WAP 解决方案。

1.4.1 Nokia WAP 解决方案

Nokia 公司研究开发了 WAP 1.0 服务器系统，并于 1999 年 12 月 15 日开始该服务器的全球销售。Nokia WAP 1.0 服务器是一种让用户在移动状态下安全地访问 Internet 资源的产品，目前已有数千家公司和软件开发商试用过该服务器系统，效果十分成功，在一定程度上可以增强企业的市场竞争力。为了满足软件开发商的需求，Nokia 还在 Internet 上推出 Nokia WAP 1.0 服务器软件免费试用版及新版 Nokia WAP 工具包。

Nokia WAP 服务器是一个面向移动通信应用的开放式服务器平台，使得用户能在移动环境中充分借用 Internet 的资源 and 功能，并可使用户在无线电通信网络、Internet 或企业内



部网之间获取数据及与客户进行交流时，能够保持对端到端安全性的有效控制。Nokia 的 WAP 服务器符合 WAP 1.1 版的规范，并具有完备的安全保密功能，是第一个商业应用无线传输层保密(WTLS)体系的实例。

另外，Nokia 还与 Visa 联合推出了移动电子商务的解决方案，目的是希望金融机构和移动电话运营商能够通过移动电话，为客户提供安全的支付服务。两家将推出利用移动电话进行无线网上安全支付的标准化方案，以满足不同市场对于安全、风险管理和争议处理的要求。双方还将建立基于 WAP 标准的开放式规格，使移动用户可以通过 Internet 完成安全而有保证的支付。为了使支付更加方便，简化用户的支付过程，双方还将合作开发电子移动钱包(E-Wallet)等功能系统。

Nokia 7110 是世界上第一部支持无线应用协议的移动电话，该公司随后还推出了其他型号的 WAP 手机，如 6210、6250 及 9110 等。

1.4.2 Motorola WAP 解决方案

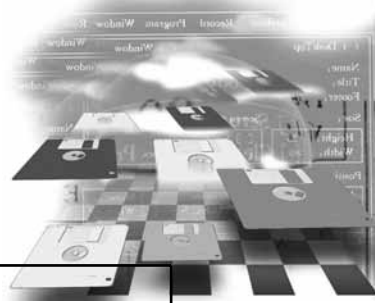
Motorola 公司一直致力于无线移动通信技术与 Internet 的最佳结合，让用户通过智能化的手持移动设备随时随地连接 Internet，获取 E-mail、金融、股票和其他信息服务。该公司以 MIX & My Sphere Internet 连接方案为核心，发布了一个完整的端到端的 WAP 解决方案。1999 年 11 月 16 日，Motorola 还公布了针对中国市场的 WAP 解决方案。同时，该公司还在中国推广“掌中网”的概念、技术和终端产品，即“Motorola 无线 Internet”一揽子方案。

“掌中网”的各种终端产品通过 GSM、CDMA、FLEX 等平台，不仅可以用于通信，也可以用于通过无线电通信方式获得 Internet 的个性化信息服务，并具有很好的移动性、便携性及实用功能，而且还能够支持、交互式的信息应用与电子商务。

Motorola 支持 WAP 的手机有 A6188、2000www、V8088 等型号。相信随着技术的发展，Motorola 还会推出更多、更先进的 WAP 手机。

1.4.3 Ericsson WAP 解决方案

Ericsson 一直关注无线技术与 Internet 的真正结合，使移动用户能够随时上网。1999 年 6 月，Ericsson 公司推出了全球第一个符合 WAP 1.1 的端对端 WAP 服务系统，随后便向多家营运商交付了几十套 WAP 服务系统。Ericsson 致力于向用户尤其是蜂窝系统运营商、服



务商和相关企业提供各类无线 Internet 解决方案，以促进无线数据通信事业的发展，增强这些客户的生产力及市场竞争能力。




除了进一步扩大市场和出售移动 Internet 解决方案之外，Ericsson 还与一些关键领域中的诸如客户/服务器应用软件和操作系统、网络内容提供、接入系统集成及分销方面的行业领先者建立伙伴关系，甚至通过战略性投资进行收购，以加强无线 Internet 解决方案的推广和影响。同时，Ericsson 公司向无线 Internet 开发商提供应用软件开发工具和建筑群，以开发第三方的解决方案。

Ericsson 所提供的接入技术及产品包括专为 ISP 设计的运营商级的 Tigris 多业务接入平台，可以提供高密度、高性能和高容错能力的网络接入功能；还包括迈向第三代移动通信的基石，也就是通用分组无线电业务(GPRS)，该业务是新型的分组数据承载业务，数据传输速率高达 115kb/s；另外还包括第三代移动通信技术，即宽带码分多址(WCDMA)，其数据传输速率高达 384kb/s，在局域网内甚至可高达 2Mb/s。

Ericsson 在中国推出的第一部内置 WAP 功能的手机为 R320sc，另外还相继推出了 R380 等型号的 WAP 手机。

1.4.4 常见 WAP 手机

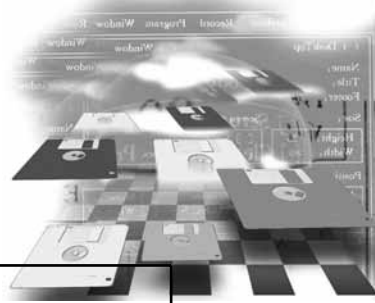
上面我们介绍了三个比较著名的 WAP 解决方案，这里我们集中介绍一下目前比较常见的支持 WAP 技术的手机。手机名称、型号、样图及性能列在表 1.1 中，仅供大家参考。

表 1.1 常见 WAP 手机		
生产厂家及型号	外观图	主 要 性 能
爱立信 R380		<ul style="list-style-type: none">• GSM900/GSM1800 双频自动切换• 红外线接口，内置 EPOC 操作系统• 具有记事簿、地址簿、闹钟、日历、计算器、语音信息记录等功能• 支持 WAP，可直接上网浏览基于 WAP 的 Internet 网站内容
爱立信 R320		<ul style="list-style-type: none">• GSM900/GSM1800 双频自动切换• 内置数据通信功能、语音备忘录• 全面支持中文信息，中文短信息服务• 支持 WAP，可直接上网浏览基于 WAP 的 Internet 网站内容
诺基亚 7110		<ul style="list-style-type: none">• GSM 900/1800 双频自动切换• 支持 WAP，可直接上网浏览基于 WAP 的 Internet 网站内容• 数据服务和信息服务功能



续表

生产厂家及型号	外观图	主要性能
摩托罗拉 T2288		<ul style="list-style-type: none"> • GSM 900/1800 双频自动切换 • 支持 WAP，可直接上网浏览基于 WAP 的 Internet 网站内容 • 全中文输入、收发中文短信息等 • 具备免提耳机、电话本、电话会议等功能 • 拥有可以显示 5 行文字的点阵图形屏幕
西门子 3568i		<ul style="list-style-type: none"> • GSM900/GSM1800 双频自动切换 • 红外线接口，内置传真/数据调制解调器 • 6 行中文显示，高清晰度大显示屏 • 支持 WAP，可直接上网浏览基于 WAP 的 Internet 网站内容
西门子 3518i		<ul style="list-style-type: none"> • GSM900/GSM1800 双频自动切换 • 内置传真/数据调制解调器 • 支持 WAP，可直接上网浏览基于 WAP 的 Internet 网站内容
诺基亚 6210 手机		<ul style="list-style-type: none"> • 支持中文短消息及中文输入(拼音、笔划)，内装四组游戏 • 可与普通计算机或笔记本电脑传输资料 • 可快速传输数据和图像，最高传输速度 43.2kb/s • 具有多功能电话簿，可记载姓名、电话、传真号码、住址等 • 支持 WAP，可浏览 WAP 网站信息
爱立信 A2618S 移动电话		<ul style="list-style-type: none"> • 中文双频，拥有 WAP 服务、语音拨号应答等先进性能 • 允许用户根据不同场合调节电话的配置文件 • 支持 WAP，可浏览 WAP 网站信息 • 开机和关机伴有声音和动画效果 • 显示屏可以显示 4 行文本。 • 内装有三种游戏：Tetris、Erix 和 Maze
摩托罗拉 L2000WWW	(图略)	<ul style="list-style-type: none"> • GSM900/1800 和 PCS1900 三种频段切换 • 来电振动功能，全中文键盘输入，内置时钟、日历等 • 具有中文短信的接收/发送功能 • 声控菜单、语音拨号、内置红外接口 • 能存储已拨和接收电话各 10 条 • 手机内可存储 100 个姓名和号码 • 可管理 SIM 卡中的 255 个姓名和号码，需 SIM 卡支持 • 支持 WAP，可直接上网浏览基于 WAP 的 Internet 网站内容
飞利浦 989	(图略)	<ul style="list-style-type: none"> • GSM1900/GSM1800 双频自动切换 • 内置数据通信功能及 Modem、语音备忘录 • 50 个反映情绪的“情感图标”，支持声控拨号及操作 • 20 个铃声选择及自创铃声，来电振动，20 秒录音留言 • 支持 WAP，可直接上网浏览基于 WAP 的 Internet 网站内容



续表

生产厂家及型号	外观图	主 要 性 能
松下 GD93	(图略)	<ul style="list-style-type: none">• 属于 GPRS 手机，利用 GPRS 的宽带，使得无线传输图像、音乐等成为可能• 结合蓝牙技术(下节介绍)，可极大地方便用户进一步拓展有关功能• 自动双频 900/1800MHz• 支持 GPRS，可直接上网浏览基于 WAP 的 Internet 网站内容• 大屏幕显示• 可存储 MP3 音乐，具备录音、自编铃声，并可听声辨人• 具有来电振动、日历、闹钟等功能，支持 STKSIM• 内置 MODEM

1.5 手机蓝牙技术简介

WAP 技术为移动电话上网开辟了一片广阔天地，但新近出现的蓝牙(BlueTooth)技术为语音、文字及影像的无线传输大开方便之门，为移动通信带来了新的视野。目前 BlueTooth 技术还存在不少问题，尚有许多细节亟待解决，但该技术对无线通信的影响却非常显著，而且意义深远。下面我们就简要介绍一下蓝牙技术。

BlueTooth 技术是以 Ericsson 公司为主，联合 Intel、IBM、Nokia、Toshiba 等公司组成的特别兴趣小组 SIG(Special Interest Group)所共同创立的。目前 Motorola、Simens、Alcatel、Canon、Acer 等近 2000 家公司已经加入到 BlueTooth 技术行列中来，而且加入的公司数目还在不断地增长，或许将来绝大多数的产品都会植入 BlueTooth 来实现无线电连接。

BlueTooth 原为丹麦 1000 多年前某个皇帝的名字，他为统一四分五裂的瑞典、芬兰和丹麦做出了不朽的功劳。瑞典 Ericsson 公司把即将成为全球通用的无线电技术命名为 BlueTooth，也许有这种技术必将一统无线电技术标准天下的含义。

BlueTooth 是无线数据和话音传输的开放性标准，主要解决短距离的无线连接，一般为 10 厘米到 10 米的范围，如果增加功率或是加上相关外部设备如专用的放大器(Optional Amplifier)等则可达到 100 米的距离。作为一种新的无线电通信协议，BlueTooth 有可能取代 IrDA 红外线技术，并将取代现有的串口(Serial Port)。

BlueTooth 工作时采用 2.402~2.480GHz 的高频无线电频率，中心频率为 2.45GHz，比现有的 GSM1800 还要高。在发射机频率为 1MHz 时，有效的 BlueTooth 数据传输速率是 721Kb/s。相对 WAP 而言，BlueTooth 规范化了更为具体的硬件及频率应用等内容。虽然说 WAP 也是无线电通信协议，但更准确地讲，它应当是无线电访问协议，或者是无线上网



协议。WAP 规范了网页的浏览格式及通信协议,如 TCP/IP 等,而 Bluetooth 协议则能使包括蜂窝电话、掌上电脑、笔记本电脑、相关外设和家庭 Hub 等众多设备之间进行信息交换。Bluetooth 结合电路开关和分组交换机形成的宽带协议,尤其适用于语音和数据的传输。传输声音时每个声道可支持每秒 64KB 的同步(语音)链接。同时,其异步信道支持任一方向上高至每秒 721KB 和回程方向每秒 57.6KB 的非对称链接。因此,Bluetooth 技术可以足够快地应付蜂窝系统上非常大的数据传输速率。单芯片的植入 Bluetooth 的电子元件的实际尺寸比西服的纽扣还要小,因而可以很方便地应用于电子产品中。

Bluetooth 技术的主要应用包括以下几个主要方面:

(1) 支持 Internet 接入。内置 Bluetooth 芯片的笔记本型计算机或手机等,不仅可以使公用电话交换网 PSTN(Published Swithed Telephone Network)、综合业务数字网 ISDN(Integrated Services Digital Network)、局域网 LAN(Local Area Net)、包含非对称数字用户线路 ADSL(Asymmetrical Digital Subscriber Loop)在内的 x 数字用户线路 xDSL 等进行 Internet 连接,而且可以使用蜂窝式移动网络进行 Internet 连接。

(2) 灵活切换连接方式。在母机有效发射范围内,比如在家里,内置 Bluetooth 芯片的手机可以当作无绳电话使用。这一点对中国用户很重要,因为当作无绳电话使用时能够不被双向收费,可以节省手机费用。当用户离开母机发射的有效范围时,作为无绳电话使用的手机就会自动切换到无线网络基站上。

(3) 手机与计算机无线连接。目前手机与计算机的连接多数是通过 IrDA 红外线或 RS232 串口线来实现的,现在 Bluetooth 技术可以取代这些连接方式。Bluetooth 技术不仅应用方便而且资料传送速度也很高,同时可以简化手机连接器,甚至将来也有可能去掉连接器。

(4) 无线免提。采用 Bluetooth 技术连接的手机可以实现无线免提工作,这对某些领域来说是十分有用的,比如汽车车载移动电话、多人视频会议等。

(5) 同步资料传输。不论是在办公室或是在家里,用户都可以通过 Bluetooth 及其相应程序,实现计算机、笔记本电脑、手机、PDA 等信息的同步传输。这里传输的信息不仅可以是文字,而且可以是图片、影像。使用带有 Bluetooth 功能的数码相机,用户拍摄完成后,可将影像传至手机进而可以传送其他任何目的地。

(6) 大面积实现数据共享,使得用户办公更加简易。利用 Bluetooth 技术,手机、计算机、PDA、打印机、数码相机、MP3 播放器等都可以实现语音、文字、图像、文件等信息的互相传输,实现大面积的数据共享,大大简化办公设施,使得用户办公更加简单、容易。



虽然说 Bluetooth 技术可以在上述方面获得很好的应用,但由于该技术目前还存在一些限制,甚至是缺陷,所以 Bluetooth 技术还需要进一步的研究。当前 Bluetooth 技术存在的问题主要包括如下几个方面:

(1) 频段问题。Bluetooth 技术采用的频段,目前 IEEE802.11 协议也同样采取这一频段。如果用户带着一台 Bluetooth 的设备来到一个装备 IEEE802.11 无线网卡的局域网环境,二者就会发生干扰,目前还没有很好的解决办法。

(2) 全方位问题。Bluetooth 具有全方位的特性。如果用户所处环境中有多数的采用 Bluetooth 技术的设备同时工作,那么这些设备就会互相影响,即便采用靠近或者瞄准的方法都可能仍不能使它们正常工作。相应地,这也会影响设备的工作效率。

(3) 安全性问题。Bluetooth 技术虽然优于 IrDA 的多向性传输,但它的安全性有些问题,可能会发生信息误传或被截取的情况。为避免这一问题,需要每个采用 Bluetooth 技术的设备都要有不同的频度,这在频度范围有限的情况下,目前还难以完全解决。

(4) 一对多点。Bluetooth 具有一对多点的数据交换能力,也就是说,一台设备发出数据,周围好多设备都可以接收。因此,Bluetooth 技术需要安全系统来防止未经授权的访问。不过目前这一点并未得到很好的解决。

(5) 速度与带宽问题。Bluetooth 的通信速度为 750Kb/s,带宽仅 1Mb/s。然而,现在带宽 4Mb/s IR 端口的产品比比皆是,而且 16Mb/s 的扩展也已在业界获得批准,所以 Bluetooth 技术的速度与带宽需要大幅度提升。

本章小结

本章在介绍 WAP 论坛的基础上,介绍了 WAP 的组成及主要特点,分析了 WAP 的应用优势与劣势,介绍了 WAP 应用现状,并对 WAP 应用的未来做了简单预测。同时,还介绍了 Nokia、Motorola 和 Ericsson 三家著名移动通信设备公司的 WAP 解决方案,并对新近出现的手机蓝牙技术做了简单介绍。虽然说本章都是知识性的、铺垫性的,但深入理解本章涉及的几个基本概念,对于大家顺利地学好后续内容必定会大有裨益的。为此,我们这里总结性给出本章涉及的几个概念,供大家参考。

WAP——无线应用协议 WAP(Wireless Application Protocol)也称为无线应用程序协议,是在数字移动电话、Internet 及其他个人数字助理机 PDA、计算机应用之间进行通信的开放



性全球标准。WAP 由一系列通信协议组成,主要包括无线传输协议 WTP(Wireless Transaction Protocol)、无线数据报协议 WDP(Wireless Datagram Protocol)、无线会话层协议 WSP(Wireless Session Protocol)、无线传输安全层协议 WTLS(Wireless Transport Layer Security)及 HTTP 接口等。

GSM——全球数字移动电话系统(Global System for Mobile communications)。

CDMA——码分多路访问(Code Division Multiple Access)系统。

微浏览器——与 PC 上使用的浏览器软件类似,微浏览器(MiniBrowser)是用户通过手机等移动通信设备,或者通过 PC 模拟手机上网访问 Internet 所采用的浏览器软件。它采用标准的通信模式进行上网浏览,可以全面实现无线网络信息服务。

WML——无线标记语言 WML(Wireless Markup Language),是一种基于扩展标记语言 XML(Extension Markup Language)的语言,是 XML 的子集,可用来显示文字、图像等数据。

WMLScript——WML 的脚本语言,在 WML 的基础上增加了一些功能,如实现对用户输入数据的有效性进行检查等,这一方面增强了 WML 的浏览和表示功能,另一方面对用户的操作也给予了更加灵活和智能的处理。

WAE——WAP 标准所定义的无线应用环境 WAE(Wireless Application Environment),能够让设计人员开发独立于设备的用户界面,并可使用 WML 脚本 WMLScript 的 WAP 编程语言,把可执行的逻辑嵌入到移动终端中。

WTA——WAP 标准定义的无线电话应用 WTA(Wireless Telephony Applications),可使 WAP 很好地与目前电信网络中现存的各种先进电信业务相结合,并可通过使用浏览器方式的用户接口,让移动用户直接应用各种智能网业务。

BlueTooth——指无线数据和语音传输的开放性标准,属于一种新的无线电通信协议,主要解决短距离的无线连接,一般为 10 厘米到 10 米的范围,如果增加功率或是加上相关外部设备如专用的放大器(Optional Amplifier)等则可达到 100 米的距离。该技术目前不甚成熟,仍在迅速发展之中。



第2章 WAP 原理、架构与开发工具包

与超文本传输协议 HTTP(HyperText Transfer Protocol)一样, WAP 协议也是一种通信标准, 规定了 Web 服务器与客户浏览器之间通信的方式、交互的方式和一系列规范。在进一步学习 WAP 编程知识之前, 我们先来了解 WAP 协议的组成、WAP 工作原理和系统架构、WAP 测试环境的建立方法以及常用 WAP 开发工具包等内容。

2.1 WAP 协议层组成及内容

WAP 由一系列协议组成, 同时还引用了许多 Internet 协议, 比如 IP、UDD、XML 等, 并为基于 HTTP 和 TLS 的 Internet 标准协议预留了空间。目前, Internet 技术主要是针对 PC 设计的, 能够支持在可靠度高的数据网上进行宽带连接, 然而, 像移动电话这种大众化的便携式无线装置在数据、信息等方面的处理能力上根本无法与 PC 相提并论。这是因为, 第一, 移动通信产品主要是移动电话, 其中央处理器 CPU 的速度较慢、内存较小、电力有限、显示屏较小、按键数量较少、输入方式有限; 第二, 由于移动通信系统本身的原因, 导致移动网络带宽较窄、稳定性较低、服务内容比较简单, 不适合用于接收 Internet 信息。所以, WAP 针对移动网络的需要, 为了适应无线电通信的特殊环境进行了特别设计和优化。

那么, 移动网络有哪些需要, 或者说 WAP 需要有哪些服务内容呢? WAP 的服务内容主要包括 World Wide Web 信息浏览、E-mail 收发、IRC 网上实时聊天和 Newsgroups 新闻组讨论等。WAP 只要求移动电话和 WAP 代理服务器的支持, 而不要求现有的移动通信网络协议作任何的改动, 所以 WAP 能同时适用于 CDMA、DETC、GSM、IMT-2000 等多种不同的移动通信系统。WAP 协议堆栈的设计也力求使所需带宽最小化, 并对各种网络技术和提供服务提供广泛支持, 包括短消息服务 SMS、USSD、CDPD 等。而且, WAP 建立了一个比较松散的层次结构, 每层的开发独立于其他层, 这样就比较容易能够引入新的传输协议和

服务类型。如图 2.1 所示，我们给出了 WAP 协议层的组成示意图。其中主要的协议层解释如下：

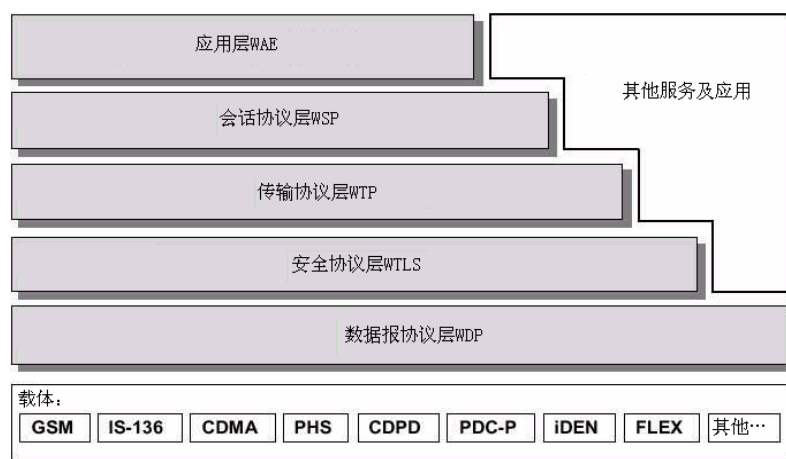


图 2.1 WAP 协议层的组成

(1) 应用层。即无线应用环境 WAE(Wireless Application Environment)，它是基于 WWW 和移动电话技术而建立的一种通用应用环境，其基本目的是构建一个可共同操作的环境，以便允许操作人员和服务供给者创建适用于不同无线平台的应用与服务。WAE 提供了一个微浏览器，包含有下列功能：

- 解释并执行使用 WML 语言编辑的 WAP 网页；
- 包含 WML 脚本即 WMLScript，并能解释和执行采用该脚本语言编写的网页；
- 支持无线电话技术应用，包括电话技术服务 WTA 及其程序设计界面 WTAI；
- 定义了一组明确的数据格式，包括图像、电话本记录和日期信息等的的数据格式。

(2) 无线会话层。无线会话层协议 WSP(Wireless Session Protocol)向两个对话服务提供一致接口的 WAP 应用层。其一在 WTP 层上操作的连接导向服务，其二是在安全或非安全数据包服务上操作的非连接服务 WDP。无线会话协议当前由与浏览应用相匹配的服务组成，通常简记为 WSP/B，它提供下列几项功能：

- 支持在压缩的超空间编码中的 HTTP/1 的功能和语义；
- 支持长久对话状态，以及通过对话移动暂停和恢复；
- 支持可靠或不可靠数据的普通设备的连接与访问；
- 支持协议特性流通。由于 WSP 体系的协议需要较长的反应时间，所以对低带宽载体网络的应用进行了优化，从而使 WSP/B 设计得允许 WAE 代理把 WSP/B 客户连接到 HTTP



服务器。

(3) 传输协议层。无线传输协议层 WTP(Wireless Transaction Protocol)在数据包服务的顶端运行,并提供适合在“瘦”客户即移动网络上执行的普通事务服务,并可对移动终端进行优化,主要提供以下功能:

- 三个级别的传输服务:不可靠单向请求、可靠单向请求、可靠双向请求与答复;
- 用户对收到信息的确认;
- 对超频带数据的确认;
- 旨在减少传送信息数量的 PDU 串联延迟;
- 异步传输服务。

(4) 安全协议层。无线传输安全层协议 WTLS(Wireless Transport Layer Security)是基于工业标准传输层安全协议的协议,它在安全传输协议 SSL 的基础上针对 WAP 传输所用的低带宽通信信道进行了优化,主要为数据传递提供下列功能和服务:

- 保证数据在终端和应用服务器间稳定、准确地传送;
- 保证数据在终端和应用服务器间传输的保密性,避免数据传输中的截取、窃听;
- 保证终端应用服务器的真实性;
- 对不能顺利通过核对的数据进行检测,如果必要则驳回数据,使对方重新发送;
- 保证终端之间的通信安全。

(5) 数据报协议层。无线数据报协议 WDP(Wireless Datagram Protocol)用于传输数据,发送和接收消息。它可以向 WAP 的上层协议提供服务支持,并保持通信的透明性,同时能够独立运行下部无线网络。在保持传输接口和基本特性一致的情况下, WDP 采用中间网关可以实现全局工作的互用性,从而实现无线数据的顺利传输。

了解了 WAP 协议层组成及其内容,接下来我们就可以认识 WAP 工作原理及其系统架构了。有关这方面的内容,我们集中在下一节介绍。

2.2 WAP 工作原理及系统架构

我们先来讲解 WAP 的工作原理,然后再与 Internet 系统架构相比较,来分析 WAP 系统的网络架构。

2.2.1 WAP 工作原理

作为开放性的全球规范，WAP 可以使移动用户利用无线电设备方便地访问或交互使用 Internet 应用信息和服务。前文述及，在 Internet 中，一般的协议要求发送大量的主要基于文本的数据，而标准 Web 内容很难在移动电话、寻呼机之类移动通信设备的小尺寸屏幕上显示。同时，在用户单手持机的情况下，屏幕间的内容切换也很不方便，并且 HTTP 和 TCP/IP 协议也没有提供针对无线网络的非连续的信号覆盖、长时间的延时以及对有限带宽所进行的优化处理。在 Internet 中，HTTP 协议不是以压缩的二进制方式，而是以效率不高的文本格式发送标题和命令。因此，如果在无线电通信服务中使用普通 Internet 协议，则会导致速度慢、成本高且难以大规模应用等问题，而且无线电传输的延时还会造成其他一些问题。

为了解决此类问题，WAP 进行了很多优化处理。比如，利用二进制传输经过高度压缩的数据，并对长延时和中低带宽进行优化。WAP 的会话功能可以处理不连续覆盖的问题，并能自动地在 IP 不可用时改用其他优化协议来进行各种信息传输。通过使用 WML 语言编写网页，WAP 还解决了 Internet 页面不能在移动通信设备上显示的问题。运用 WML 编辑的网页可在手机的微浏览器上产生按钮、图示及超链接等功能，并可提供信息浏览、数据输入、文本和图像显示、表格显示等功能，大大减小了在移动设备上浏览网页内容的复杂程度。

另外，WAP 通过加强网络功能来弥补便携式移动设备本身的缺陷，工作时尽可能少地占用移动通信设备的资源，比如 CPU、内存等。与 Web 对 Internet 的作用一样，WAP 在应用层上隐藏了 GSM 的复杂性，给用户提供了类似于普通 Web 页面的友好性。WAP 还通过使用类似于 JavaScript 的脚本语言 WMLScript，来使移动通信设备先将信息进行处理后再发给服务器。WAP 还通过无线电话应用 WTA 来实现呼叫控制等诸多电话功能。

WAP 标准下的移动终端均配备了一个微浏览器，该浏览器采用了一种类似于卡片组的工作方式。用户可以通过卡片组来浏览移动网络运营商提供的各项 Web 业务。工作时，移动终端用户首先选择一项业务，该业务会将卡片组下载到移动终端，然后用户就可以在卡片之间往返浏览，并可进行选排或输入信息，以及执行所选择的工作等。而且，浏览到的信息可以高速缓存，以便供以后使用，卡片组也可以高速缓存并可做成书签以备快速检索之用。该浏览器同时还对电子名片、日历事件、在线地址簿和其他类型内容的格式提供了

相应支持。

2.2.2 Internet 与 WAP 的系统架构

为了说明 WAP 的系统架构，我们先来分析一下普通 Internet 中 Web 服务器的工作方式和工作原理。

在图 2.2 所示的 WWW 模型中，客户向 URL 所指定的 Web 服务器发出一个请求，Web 服务器收到该请求后，经处理即返回相应的内容至客户端。这个过程中，双方是按照 HTTP 协议进行交互的。客户端发出一个以 HTTP 开头的 URL 请求时，Web 服务器端处理该请求的程序可能是 CGI 程序、静态网页，也可能是 Servlet 程序，甚至可能是其他服务器端的程序，但它们都是以 HTML 格式将相应的内容返回给客户，这样，客户就可以在浏览器上看到返回的具体内容。

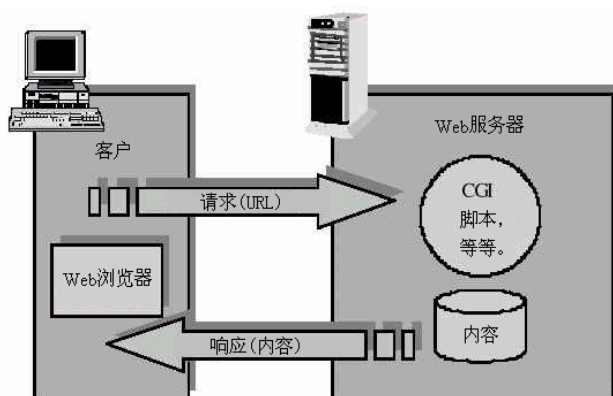


图 2.2 WWW 模型

WWW 模型(图 2.2)还同时说明了建立普通应用环境所需的必要配置，主要包括以下几个方面：

- (1) 标准命名模型。WWW 上所有的服务器和内容都是通过 Internet 标准的信息指定方法进行命名的。
- (2) 内容键入。主要指 URL 的键入，WWW 为此定义了若干特定的类型，允许网络浏览器在此基础上进行正确的处理。
- (3) 标准内容格式。所有的网络浏览器均支持一组标准的内容格式，包括超文本标记语言 HTML、Java 描述语言以及其他格式。
- (4) 标准协议。标准网络协议允许任何网络浏览器连接到任何网络服务器上。WWW 体

系中最常用的协议是 HTTP 协议。WWW 的这种基本结构可使用户方便地运行、获取第三方的应用软件及内容服务，并可使开发人员方便地为广大客户创建特定的应用软件和内容服务。

下面我们再来分析 WAP 的网络架构。WAP 网络架构由 3 部分组成，即 WAP 网关、WAP 手机和 WAP 内容服务器。其中，WAP 网关起着“翻译”协议的作用，是联系 GSM 网与 Internet 的桥梁；WAP 内容服务器可以存储大量信息，以供 WAP 手机用户来访问、浏览和查询等；WAP 手机为用户提供了上网用的微浏览器及信息、命令的输入方式等。图 2.3 所示就是 WAP 模型的基本网络架构。当用户从 WAP 手机键入想要访问的 WAP 内容服务器的 URL 后，信号经过无线网络，以 WAP 协议方式发送请求至 WAP 网关，然后经过“翻译”处理，再以 HTTP 协议方式与 WAP 内容服务器交互，最后 WAP 网关将服务器返回的内容压缩、处理成二进制流，并返回到客户的 WAP 手机屏幕上。编程人员需要解决的问题是编写 WAP 内容服务器上的程序或者说 WAP 网页，有关 WAP 网页的编写方法我们后面会详细介绍的。

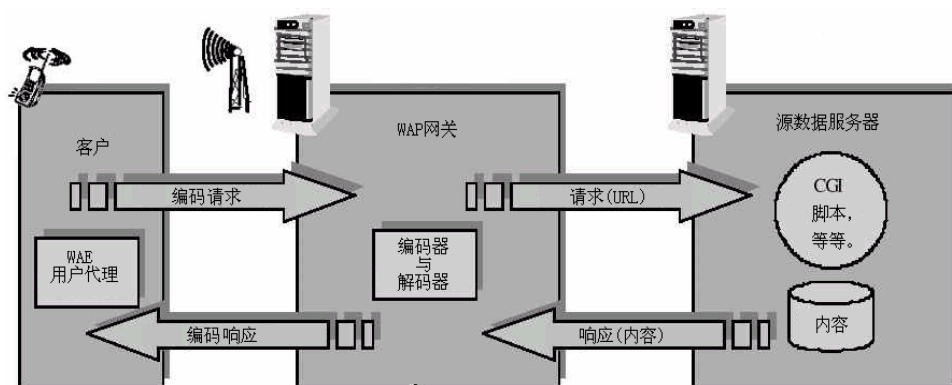


图 2.3 WAP 模型

与 WWW 模型一样，WAP 也定义了一组旨在促进移动终端与 WAP 内容服务器之间通信的必要配置，主要包括以下几个方面：

- (1) 标准命名模型。WAP 与 WWW 一样，其服务器和内容都是通过 Internet 标准的信息指定方法进行命名的。
- (2) 内容键入。主要指 URL 的键入，WAP 建立了与 WWW 一致的内容形式和类型，允许 WAP 用户代理在此基础上进行正确的处理。
- (3) 标准内容格式。WAP 基于 WWW 技术，所用微浏览器也支持一组标准的内容格式，

包括 WML 及其脚本语言、图像、日历信息、电子名片甚至涨价幅度等的格式。

(4) 标准协议。WAP 网络协议允许手机中的微浏览器通过 WAP 网关连接到 WAP 内容服务器上，满足了移动终端与网络服务器之间传输信息的要求。

关于 WAP 的网络构架，我们还有几点需要强调说明：

(1) WAP 的客户端一般都是 WAP 移动设备，比如最常见的手机。它们通常由不同的公司生产，各自具有不同的特点，所以对 WML 和 WMLScript 的解释也有所不同。因此，将来具体开发时，还要参考各个厂家提供的技术资料，综合考虑不同品牌、型号的 WAP 手机特点，如 Nokia 7110、Ericsson R380、Motorola L2000www 等等，以使开发的网页及应用能为绝大多数的手机所接受。

(2) 一般来说，用户都希望 WAP 手机的屏幕足够大，分辨率足够高，否则浏览网页时不是太方便。然而由于不同型号 WAP 手机的屏幕大小并不完全一样，所以开发人员需要考虑针对不同的手机制作不同的网页，或制作含有多种选择的网页，以便使得客户在浏览时同样感到合适、方便和快捷。而且，由于各种手机对 WML 和 WMLScript 的支持情况不太一样，就像 Internet Explore 和 Netscape 之间的情况一样，所以开发时更是要考虑不同手机的特殊情况。

(3) WAP 手机上网中的安全性问题在开发中也需要考虑，如散射在空间中的电波会同时把用户的口令、密码等重要信息同时散射在空间中。本书后面将专门讨论 WAP 手机上网的安全性问题。

(4) 网关在 WAP 系统的整个构架中有着十分关键的作用，它是连接客户和服务的桥梁。网关可以在电信局一端(见图 2.4)，也可以和网络服务器集成在一起(见图 2.5)。

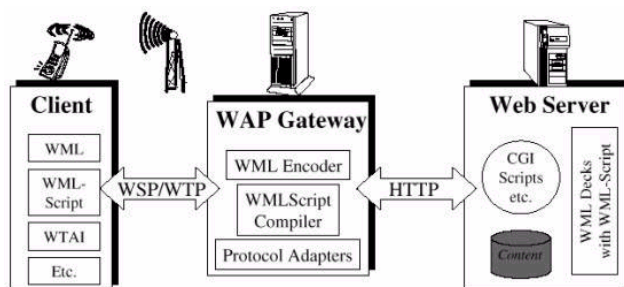


图 2.4 网关在电信局一端时的结构图

具体开发中，要根据这两种情况采用和实施不同的设计思想及方法。一般来说，如果网关在电信局一端，那么在服务端开发应用程序时与以前的网页开发没有太大的区别。只

要符合通用网关接口 CGI(Common Gateway Interface)标准,无论是采用 Perl、C、C++、Basic、Dephi 或是专门的 ISAPI、Serverlit 等,开发方法都是一样的。这种情况下开发人员拥有比较大的自由。

如果网关和网络服务器集成在一起,那么开发时就要考虑应用逻辑及其相关问题,并在此基础上,综合利用 CGI 标准和常用开发工具,设计与开发 WAP 网页及应用。

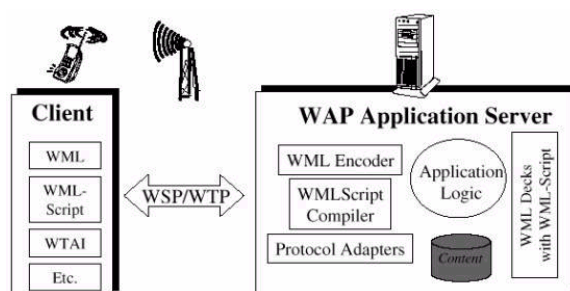


图 2.5 网关也可以和网络服务器集成在一起

2.2.3 WAP 与 Internet 的比较

我们学习 WAP 工作原理和架构的目的是为了以后深入掌握 WAP 编程,因此,我们需要了解 WAP 编程中究竟有哪些开发工作。如图 2.6 所示,我们同时列出了 Internet 和 WAP 的系统层次结构。图的左部是 Internet 的各个层次,右部是 WAP 的各个层次。比较一下可以看出, WAP 的结构层次要比 Internet 复杂得多。虽然 WAP 的整个结构层次比较复杂,但是,由于它底层的大部分工作都是由电信部门和移动电话公司通过相关软硬件系统来完成的,因此总的来说,我们只需要关心 WAP 应用层的开发工作。这一点也与 Internet 中 HTML/JavaScript 层的开发工作相类似。

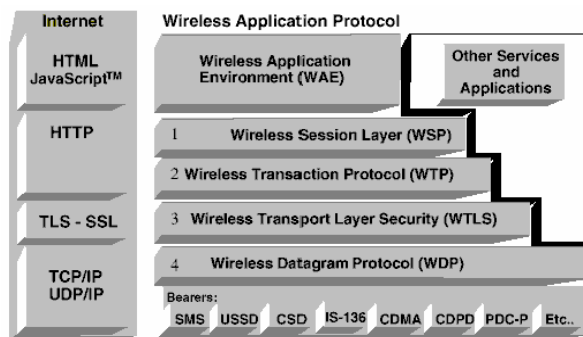


图 2.6 WAP 与 Internet 系统结构层次比较

另外, 在我们进行 WAP 开发时, 由于 WAP 各部分的协议与 Internet 上的协议有着一定的对应关系, 所以我们可以使用现有的 Internet 服务器来实现 WAP 相关服务。

如图 2.7 所示, WAP 的 WML 与 Internet 的 HTML 对应, 所以用户使用 WML 开发 WAP 网页时, 可以像使用 HTML 语言来开发 Internet 网页一样来工作。再如, WMLScript 与 JavaScript 对应, 开发时具体的处理方法也基本一样。其他对应的开发项目还有: WTAI 与移动网络、WSP 与 HTTP、WTLS 与 SSL/TLS、WTP 与移动网络/TCP/UDP/IP 等。根据这些对应项目, 在进行 WAP 开发时可参考相应 Internet 中的开发工作, 以简化设计工作, 提高开发效率。

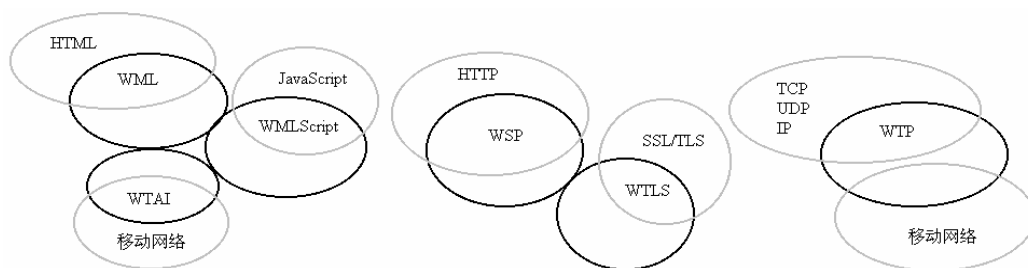


图 2.7 WAP 与 Internet 的相互关系

2.2.4 WAP 网络服务方案

目前, WAP 网络服务方案主要有 3 个, 如图 2.8 所示, 各方案情况介绍如下:

方案一: Web 服务器(Web Server)传送原有的 HTML 网页, 由 HTML 过滤器将该网页转换成 WML 格式, 然后再由 WAP 代理(WAP Proxy)服务器处理后形成二进制的 WML 数据流送往客户端, 即用户的 WAP 手机。

方案二: Web 服务器直接将 WML 网页传送到 WAP 代理服务器, 然后由服务器处理后形成二进制的 WML 数据流送往客户端。

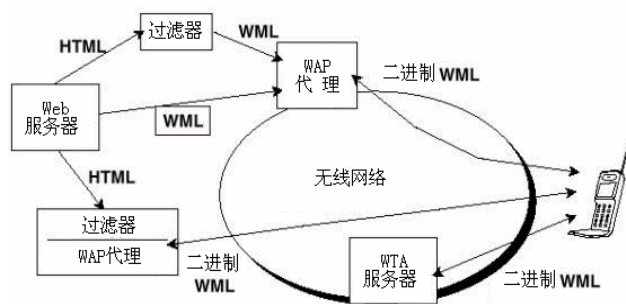


图 2.8 WAP 网络服务方案



方案三：由 WTA 服务器(WTA Server)直接将二进制的 WML 数据流送往客户端。不过这种方案用途有限，主要是用于提供电话呼叫等服务。

对于 WAP 服务供应商来说，主要考虑前两个方案。第一个方案的好处是开发者不需要对原有的网络及网页进行修改，HTML 到 WML 的转换工作可由 HTML 过滤器直接完成。不过这个方案也是有缺点的，主要表现在 4 个方面：其一，过滤器只能做有限的转换，对于比较复杂的 HTML 网页就很难全部转换成功；其二，HTML 网页远比 WML 网页复杂，所以转换后传输的效率比较低；其三，过滤器这一中间环节的增加，就整个系统而言降低了网络的工作效率；其四，增加了 HTML 服务器的负载，影响了整个系统的效率。显然，这一方案并不完全符合 WAP 服务“简短、快捷”的要求。

相比之下，第二种方案比较合适。但第二方案也存在一个问题，那就是我们不一定非得要增加 WAP 代理服务器，因为实际上我们可以利用已有的 HTML 服务器来实现 WAP 服务。特别是 WAP 手机只接收能够进行解释的二进制 WML 数据流，所以我们根本无需考虑这个数据流的提供者。这样，我们还可以进一步优化该服务方案，提高系统工作效率。

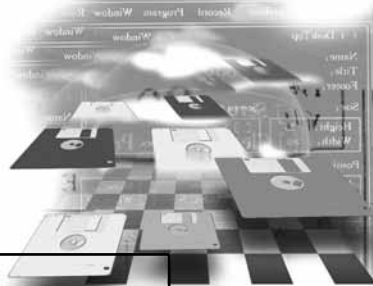
2.3 WAP 测试环境

在 WAP 编程与开发中，为了对所编网页及应用进行测试，我们通常需要建立 WAP 的测试环境。一般来说，WAP 测试环境可以从浏览器环境、模拟环境、实际环境三个方面进行建立，本节就对此作一基本介绍。

2.3.1 浏览器环境

浏览环境的建立是十分简单的。目前 Internet 上有许多站点提供有 WML 浏览器的免费下载服务。比较著名的 WML 浏览器是 Winwap(读者可从 <http://www.wapschool.com/chinese/download/winwap22.exe> 下载)以及各移动通信设备公司提供的浏览器。安装这些浏览器后，用户就可以在 Windows 系统环境下访问 WAP 站点，查看 WAP 页面。

在 WAP 的服务器端，开发人员则可以利用 Windows NT 4.0 或 Windows 2000 以及 Internet 信息服务器 IIS(Internet Information Server) 4.0/5.0 软件进行模拟。在原有的 WWW 服务子目录下再建立一个 WAP 子目录，将所有的 WML 网页放在其中，并对 IIS 进行必要



的配置。然后,在 WWW 服务器正常运转的情况下,开发人员通过在 Winwap 等 WAP 浏览器上输入 `http://localhost(本地计算机名)/wap/index.html` 的形式,即可进入 WAP 网页进行浏览测试。

这种测试环境的优点是实施起来比较简单,建设比较快,操作起来也比较简单易学。其不足之处在于,这种测试用的浏览器毕竟是 Windows 环境下的浏览器,支持大部分的 WML 标记,查看窗口的界面可以扩大和缩小,比较自由,因而所看到的测试效果与实际手机上的效果可能会有比较大的差别,而且它也不能提供编辑、编译和调试的集成环境。

2.3.2 模拟环境

用于 WAP 测试的模拟环境是通过使用移动通信设备公司所提供的 WAP 手机模拟器来实现 WML 浏览的。目前可以从各公司站点上下载的模拟器有 Nokia Toolkit、Ericsson R380 Emulator、Ericsson WapIDE、UpPhone UP.Simulator、Motorola Mobile ADK 等。

相比较来说, Nokia 和 Motorola 提供了比较完整的集成开发环境,其他两家主要提供了模拟 WAP 手机的 WML 浏览器。由于模拟器一般都提供直接的 HTML 服务连接,所以 WAP 服务器端只需要 Windows NT/2000 及 IIS 4.0/5.0 软件即可进行模拟和调试。与上面介绍的方法一样,在 WWW 服务器工作正常的情况下,通过在模拟浏览器上输入 `http://localhost(本地计算机名)/wap/index.html` 的形式,即可进行 WAP 网页的浏览测试。

虽然说这种模拟环境提供了集成环境及与 WAP 手机基本一致的模拟器,但仍难保证所用模拟器与其实际产品完全一致,尤其是没有 WAP 网关的参与,因此这是一种并不完备的检测。特别地,这种模拟环境下与无线电话应用 WTA(Wireless Telephony Application)相关的服务根本没有办法进行检测。不过,对于单纯的开发测试来说,这样模拟环境基本能够满足要求。

为便于大家下载,我们给出了几个模拟器的下载网址,如表 2.1 所示。本书所附光盘中提供了一些模拟器(免费版或限时版),读者可从中选择使用。

表2.1 3个常用模拟器的下载网址

模拟器名称	下载网址
Nokia 模拟器	<code>http://www.cwap.com.cn/webdir/wap/download/Nokia.zip</code> 或 <code>http://www.airwap.com/download/Nokia.zip</code>
Ericsson 模拟器	<code>http://www.cwap.com.cn/webdir/wap/download/ericsson.zip</code> 或 <code>http://www.airwap.com/download/ericsson-airwap.zip</code>



续表

模拟器名称	下载网址
Up 模拟器	http://www.cwap.com.cn/webdir/wap/download/upsdkW40b2e.exe 或 http://www.airwap.com/download/upsdkW40b2e.exe

2.3.3 实际环境

WAP 测试的实际环境中需要 WAP 手机、网关及服务器 3 个部分, 因此, 为了建立 WAP 测试的实际环境, 开发者需要购买一些主流的 WAP 手机, 同时使用前面介绍的方法在原来的 HTML 服务器上建立一个 WAP 专用的虚拟目录, 以建立 WAP 服务器, 然后使用现有网关或加载移动通信公司提供的相应网关, 那么只要三者都能顺利、正确地工作, 开发者就可以通过 WAP 手机对 WAP 网页及应用进行测试了。

2.4 WAP 开发工具包

随着 WAP 技术的发展和迅速普及, 许多移动通信设备公司及 WAP 发展商纷纷推出自己的 WAP 应用开发工具。为便于大家熟悉这些开发工具, 我们这里介绍目前 3 个主流的 WAP 工具包, 即 Ericsson WapIDE 2.0、Nokia WAP Toolkit 1.2 和 Phone.com UP.SDK 4.0。

虽然 Ericsson、Nokia 和 Unwired Planet(Phone.com)三家公司都是 WAP 论坛的主要成员, 但他们的开发工具包并不一致, 在很多方面都有不同。而且这三家公司都开发了自己的商用 WAP 服务器, 并且各自的服务器通常只与自己推出的开发工具包能够较密切地结合使用。这一点希望能够引起开发人员的注意。

2.4.1 Ericsson WapIDE

Ericsson WapIDE 工具包是由一系列支持设计及测试 WAP 应用的工具构成。WapIDE 的 SDK 目前只能在 Windows NT 4.0 和 Windows 95/98 环境下使用, 针对 Windows 2000 环境的 WapIDE 开发工具包, 相信不久即会面世。WapIDE 还同时提供了用于测试服务器应用的程序, 如 Perl 5.0、Tcl/Tk 以及 Xitami Web Server 等。

安装 WapIDE 时需要首先安装 IDE, 然后安装 SDK, 并同时选择安装 Xitami Web Server 等全部组件。

WapIDE 开发工具包启动后的窗口界面如图 2.9 所示。它包含有用于测试应用程序的浏览器(Browser)，用于编写应用程序的应用设计器(App Designer)和一系列服务器端工具集(Server Toolset)，比如 WML、WMLScript 编译器及语义分析器等。



图 2.9 WapIDE 开发工具包的窗口界面

从该窗口中单击“Browser”图标，即可启动 WapIDE 的浏览器，如图 2.10 所示。该浏览器可以模拟 Ericsson 多种型号的 WAP 手机，默认情况下模拟的是 R320s 手机的样子。通过该浏览器，开发者可以测试工具包中自带的网页样例，或测试自己编写的网页及应用，测试时需输入它们的 URL 地址。WapIDE 提供的样例比较典型，包括金融、股票和日程等多方面的应用，通过测试和学习这些样例，不仅可以增强我们对 WAP 编程的理解，也可以掌握一些典型 WAP 应用的开发方法。

在图 2.9 所示的窗口中单击“App Designer”图标，可以打开它的工作界面，如图 2.11 所示。

可以看到，App Designer 的窗口中集成了 3 个组件，一是 WML 编辑器，即“源程序(Source)”窗口，用于使用 WML 语言编写和设计 WAP 网页及应用；二是一个 WapIDE 的手机浏览器，即“设备(Device)”窗口，用于显示 Source 窗口中程序的运行结果；三是一个辅助窗口，即“输出(Output)”窗口，用于显示一些运行或编辑消息，辅助开发工作。

如果我们使用 Ericsson WapIDE 工具包开发 WAP 网页及应用，那么 App Designer 窗口则是我们经常使用的窗口，大部分开发工作将通过这一窗口完成。该窗口中还有一些菜单命令和工具按钮，操作都比较简单，我们就不一一叙述了。



图 2.10 WapIDE 的浏览器

💡 Ericsson WapIDE 还包括其他一些服务器工具以及与 WAP 开发相关的 WAP 网关等产品，使用时可以参考相应的说明资料，我们这里就不详细展开了。

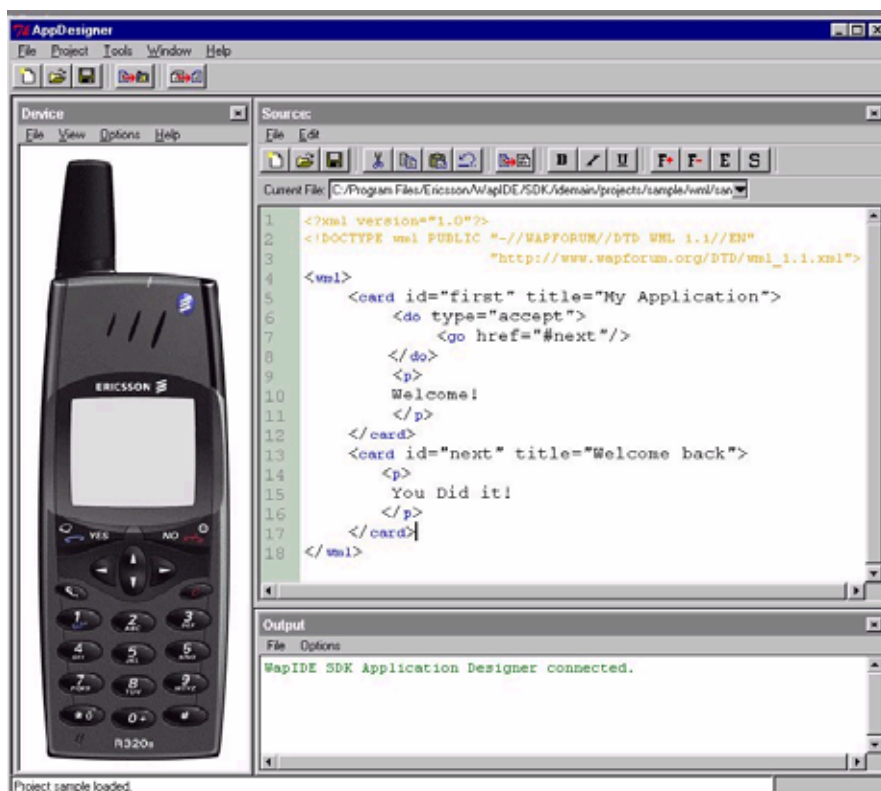


图 2.11 App Designer 的工作窗口

2.4.2 Nokia WAP Toolkit

与 Ericsson 的 WapIDE 工具包类似，Nokia 的 WAP 工具包也拥有图形开发环境、浏览器和 WML、WMLScript 的编译器。Nokia 的工具包目前只能在 Windows NT 4.0 环境中运行，而且必须先运行 Java 2 runtime 才能运行和使用 Nokia WAP Toolkit。因此，用户必须先到美国 Sun 公司的 Java 网站(<http://java.sun.com/>)去下载 Java 2 SDK 或者 Java 2 Runtime Environment(JRE)，并正确安装到所用计算机的 Windows NT 4.0 环境中，然后才可以安装 Nokia WAP Toolkit。

完全安装 Nokia 的 WAP 工具包后，可以看到它程序组里面除含有工具包本身的程序外，还含有一个工具包 IDE 以及有关 WAP、WML 和 WMLScript 的说明文档。Nokia WAP 工具

包启动后的工作窗口如图 2.12 所示, 这是一个集成化的窗口, 开发者从中可以进行 WML、WMLScript 文件的编辑、修改和测试等工作, 并可通过输入 WAP 网页或应用的 URL 地址来浏览它们的效果。

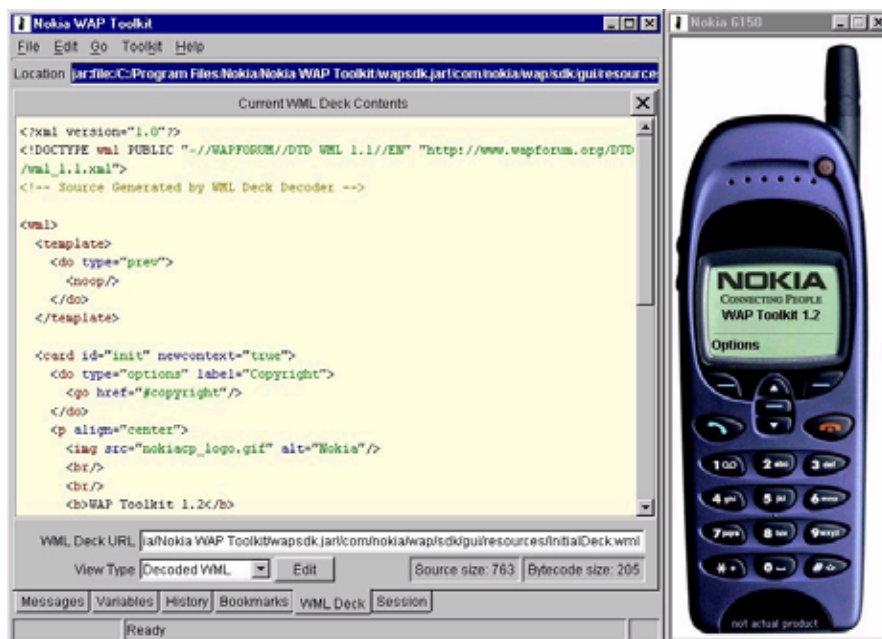


图 2.12 Nokia WAP Toolkit 的工作窗口

Nokia 还提供了一个独立的基于 WAP Server 的 Java Servlet, 集成了应用服务器和 WAP 网关两种功能。对于 WAP 服务提供商来说, 该产品更为实用, 而且维护更为系统、简练。

2.4.3 Phone.com UP.SDK

Phone.com 公司推出的 UP.SDK 产品与 Nokia 和 Ericsson 的产品有较大的区别, 虽然它可以运行于 Windows 95/98/NT 和 Solaris 操作系统环境, 但它没有提供编辑和测试 WML、WMLScript 程序的集成环境。不过, UP.SDK 提供了一系列可由 WML、Perl、C、C++、VB 等不同语言重复使用的代码库, 同时提供 Perl 和 C 语言使用的库函数, 并可用来生成 WML 文件及控制 HTTP 请求等, 而且 UP.SDK 还提供了 SSL 证书的安全性工具。因此, 使用 UP.SDK 开发, 能够大大提高用户的开发效率。Phone.com 还提供有自己的微浏览器 UP.Simulator, 如图 2.13 所示。目前, 该模拟浏览器只能在 Windows 环境中运行。

由于 UP.Simulator 是动态地与 Phone.com 开发者网站相连接的, 所以必须在用户计算机一直与 Internet 连接的状态下才可以使用, 这样为用户随时下载样例应用或直接访问 Internet



图 2.13 UP 模拟浏览器

上的 WAP 站点提供了方便。

Phone.com 的工具包除了提供有标准的 WAP 功能外，还扩展了其他一些 WAP 功能，如传真、通知等。其中，UPLINK server 就包含有一个传真管理器的产品，可以控制用户从 WAP 浏览器上直接传真的信息，Microsoft Word 文档、ASCII 文本、RTF 和 Adobe Acrobat 文档等都可以作为传真文件。而 Notification API 产品则允许用户向其他 WAP 手机客户发送异步消息，即发送通知信息。

比较而言，Phone.com 的工具包比 Nokia 和 Ericsson 的工具包拥有更多的开发功能，但其开发工具的操作界面不太“友好”，使用起来不太方便。Ericsson WapIDE 工具包操作界面的处理上有些地方用起来不是非常方便，Nokia 的工具包提供有较好的图形开发环境。因此，对于拟采用 WAP 工具包进行开发的用户而言，不妨选择 Nokia WAP Toolkit 作为开发工具。当然，为了开发适用于 Phone.com 和 Ericsson 手机的网页及应用，用户还有必要熟悉它们的工具包。

当然，目前提供 WAP 开发工具包的公司有很多，为便于大家了解这些产品，我们在表 2.2 中列出比较常见的 WAP 开发工具包以及它们的下载网址、产品功能组件情况等。

表2.2 常见WAP开发工具包的基本情况

提供商	SDK	URL	模拟器	编辑器	调试工具	例程	帮助文档
Nokia	Nokia WAP Toolkit	http://forum.nokia.com/main.html	是	是	是	是	是
Ericsson	WAPIDE SDK 2.1	http://www.ericsson.com/developerszone/	是	是			是
Phone.com	UP.SDK 4.0	http://www.phone.com/	是			是	是
Motorola	Mobile Application Development Kit (ADK)	http://www.motorola.com/MIMS/MSPG/cgi-bin/spn_madk.cgi	是	是	是	是	是
WAPMine	WAPPage 1.0	http://www.wapmine.com/Products.asp		是		是	是
WAPObjects	WAPObjects	http://www.wapobjects.com/wapobjects/en/		是	是	是	是
PWOT	PWOT WML-Tools	http://pwot.co.uk/wml/	是		是		是
Dynamical Systems Research	WAP Developer Toolkit 1.0	http://www.dynamical.com/wap/index.html	是	是	是	是	是
Perfect Solutions	CardONE	http://www.peso.de/wap_en/index.htm	是	是			

本章小结

本章介绍了 WAP 协议层的组成及相关内容，分析了 WAP 的工作原理，并在与 Internet 系统架构比较的基础上，讲解了 WAP 系统的网络架构。随后简要介绍了 WAP 编程中主要的开发工作，以及比较常用的 WAP 网络服务方案、WAP 测试环境的建立方法、常用 WAP 开发工具包等内容。

本章内容虽然不多，但主要是理论性和概念性知识，是进一步学习 WAP 编程的基础，希望读者能从总体上掌握这些内容。尤其要熟悉 WAP 测试环境的 3 种建立方法和 Ericsson WapIDE、Nokia WAP Toolkit 及 Phone.com UP.SDK 等几种常用 WAP 开发工具包的组成、特点和使用方法。



第 3 章 WAP 手机上网设置

第 3 章 WAP 手机上网设置

如果想使用 WAP 手机上网,那么使用前除了必须通过服务商开通一般的通话服务、短消息服务、数据服务外,还要完成 WAP 手机的正确设置,然后才能顺利地通过手机上网,访问各类 WAP 站点。本章将就 WAP 手机上网的准备工作和设置操作进行简要介绍。在 WAP 编程开发中,我们通常需要使用 WAP 手机进行网页或应用服务的测试,所以掌握本章内容还是比较重要的。

3.1 WAP 手机上网前的准备工作

WAP 手机上网前的准备工作并不多,主要包括 3 个方面:

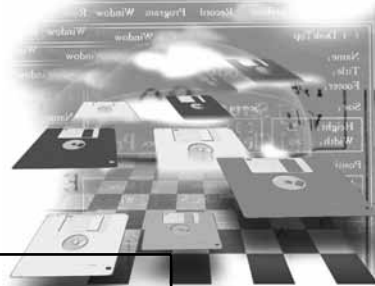
首先,用户必须要有 WAP 手机,如 Nokia 7110、Ericsson R320 等手机。当前,具有 WAP 功能的手机一般要比普通手机的价格贵,不过随着技术的发展和竞争的加剧,这类手机的价格也会逐渐降下来,或者所有的手机都会具备 WAP 功能。

其次,用户在向电信部门(如中国移动通信公司或中国联通公司)申请普通手机服务之外,还要申请 WAP 服务。当然,用户为此要支付一定的开户费和月租费。

最后,用户根据自己手机的有关说明,完成手机的上网设置,接下来就可以上网了。由于手机上网设置的内容比较多,我们集中在下一节介绍。

3.2 WAP 手机上网设置举例

由于生产厂家和手机型号不同,所以不同的手机上网时的设置任务和操作并不完全相同。具体设置时,用户应当仔细阅读手机使用说明书或相关上网设置资料,弄清楚所用手机上网时需要进行的设置项目及设置方法,按照说明书中指定的步骤和方法进行操作。下



面我们以爱立信、诺基亚、摩托罗拉、西门子等几种比较常见的 WAP 手机为例，说明上网设置的内容及操作方法。

3.2.1 诺基亚 7110 上网设置

启动并打开手机后，按照如下步骤进行操作：

- (1) 选择并进入“服务”界面。
- (2) 按动滚动移动键，进入“编辑”界面。
- (3) 按照表 3.1 的项目和设置内容，对 ISP 电话号码、连接类型、用户名、密码、IP 地址、主页网址、书签等进行设置。

设置时可能需要输入汉字，这需要切换到某一汉字输入法进行操作。用户只需连续按“#”键，即可切换和选择不同的输入方法。

表3.1 诺基亚7110上网设置项目及内容

设 置 项 目	设 置 内 容
主页	可浏览的 WAP 网址，如 wap.chnmobile.com、wap.wapfrum.com、wap.bookingall.com 等
连接类型	“持续连接”或“暂时连接”
安全保护	“关”
ISP 的接入号码	中国电信为“172”，中国联通为“1601”
IP 地址	中国电信为“10.0.0.172”，中国联通为“192.168.167.2”
鉴定类型	“普通”
数据通话类型	“模拟”
数据通话速度	“自动波特率”
用户名	“wap” (小写)
密码	“wap” (小写)

- (4) 完成上述设置后，再选择激活该设置。以后，用户只要从“功能表”中选择打开“服务”界面，然后按左边键“主页”，就可拨号连接 ISP，登录 WAP 网站，访问其中内容，或收发 E-mail 了。

3.2.2 爱立信 R320sc 上网设置

启动并打开手机后，可按照如下步骤操作：

- (1) 按动操作面板上的右箭头键，进入“WAP 服务”功能区。

(2) 再按下箭头键，进入“WAP 设置”功能区。

(3) 从中选择“Profile1”选项并进入“更名”状态，给当前设置项起一个适当的名字，具体可根据自己的喜好决定，比如“预定世界”。

表 3.2 爱立信 R320sc 设置功能及设置内容

设置项目	设置内容
通信方式	“GSM 数据”
响应时限	“150”秒
GSM 权限	“172”(中国电信)
拨号方式	“模拟”
用户名	“wap”(小写)
密码	“wap”(小写)
空闲时限	“300”秒
网关：用户标识	空
网关 IP 地址	“010.00.000.172”(中国电信)
短信息地址	无需设置

(4) 输入“预定世界”的标题，再输入相应的网址 wap.BookingAll.com，然后选择“确定”键。

(5) 接下来，再设置“图像下载”为“完全开启”模式，然后选择“确定”键。

(6) 再进入“通讯”功能区，并按照表 3.2 所示的项目进行设置。

设置时如需切换输入法，则按住“#”键 3 秒钟即可切换一种输入法。拟进行字母大小写切换时按“*”键即可。

这里的 GSM 权限即为 ISP 的上网电话号码。用户也可以选择中国联通或其他提供 WAP 服务的公司作为手机上网的 ISP，同时要设置它们的上网电话号码及网关 IP 地址。比如，中国联通的上网电话号码为“1601”，IP 地址为“192.168.167.2”。

完成上述设置后，以后用户就可以通过爱立信 R320sc 手机轻松上网了。

3.2.3 摩托罗拉 WAP 手机上网设置

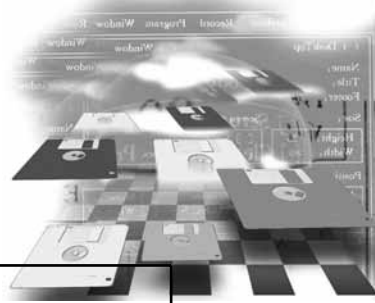
摩托罗拉 V8088、L2000www/LF200www/T2288 型号 WAP 手机的上网设置是一样的，摩托罗拉 A6188 手机的上网设置稍微复杂一点，我们先说明前面几种型号手机的上网设置内容及操作方法，然后再介绍摩托罗拉 A6188 手机的上网设置操作。

任意一款摩托罗拉 V8088 或 L2000www/LF200www/T2288 型号 WAP 手机上网时，可按照如下步骤进行设置：

(1) 在电话待机状态，按“MENU”键进入手机功能表。

(2) 从中选择“OK”键，手机即开始连接网络。

(3) 在此过程中，用户一定要按住“MENU”键，不要松开，直到屏幕显示出“浏览器菜单”后再松开。在浏览器菜单屏幕，用户可以使用箭头按键移动光标选择项目，选中的



项目将以反色显示。

(4) 将选择项移动到“7>设置网络”并按“OK”键，或者直接按“7”键，以进入设置网络功能的界面。

(5) 在“设置网络功能”界面中，用户需要设置 WAP 网关的主、次 IP 地址和 ISP 的接入电话号码等参数。可以设置的项目及设置内容如表 3.3 所示。

输入时，如果要输入句点(.)，则可按两次“1”键。如要输入斜杠(/)，则可连续按多次“0”键，直到出现该符号。有关其他特殊符号的输入，可参见手机使用说明书。

表3.3 摩托罗拉V8088、L2000www/LF200www/T2288手机上网设置

设置项目	设置内容
主页	即设置欲浏览的 WAP 网址，如 wap.chnmobile.com、wap.wapfrum.com、wap.bookingall.com 等
ISP 的接入号码	中国电信为“172”，中国联通为“1601”
IP 地址	中国电信为“10.0.0.172”，中国联通为“192.168.167.2”
超时上限	“300”秒
数据通话类型	“模拟”
数据通话速度	“9600”b/s
用户名	“wap”(小写)
密码	“wap”(小写)
连接类型	“透明”
线路类型	“调制解调器”

(6) 设置完成后，按 C 或从菜单中选择“返回”命令，再从菜单中选择“完成”回到待机状态。这时，按开关按键关机后重新打开手机，刚刚所做设置即可生效。此时选择“访问因特网”功能即可连接 WAP 站点，浏览其中的信息了。

下面我们再介绍摩托罗拉 A6188 手机的上网设置操作。与摩托罗拉其他型号的 WAP 手机不同，摩托罗拉 A6188 可以设置 1~3 个 WAP 网关代理服务器，为用户选择使用不同的 WAP 网关提供了方便。

- (1) 在手机待机状态，从主屏幕中选择“系统设置”一项，进入系统设置屏幕。
- (2) 从中选择“浏览器设置”命令，进入 WAP 功能设置屏幕。
- (3) 根据需要，从中选择“代理服务器 1”或“代理服务器 2”、“代理服务器 3”，进入相应的设置屏幕，即可分别设置代理服务器。
- (4) 从中选择各个需要设置的项目，并分别键入相应的设置内容，如 ISP 的网关 IP 地址、端口号、接入号码等。具体设置项目及内容可参见表 3.3 所示。



(5) 正确输入所有资料后,选择“确定”命令,返回到浏览器设置屏幕。

(6) 从中选择“数据呼叫信息”,进入数据呼叫屏幕。从中即可设置各项的内容,如接入电话号码、用户名、密码、通话速度、线路类型等。具体设置内容可参见表 3.3。

(7) 正确输入所有项目的设置内容后,选择“确定”命令,返回到设置屏幕,再选择其中的“确定”命令,进入待机状态。

(8) 关闭并随后重新打开话机,以使设置的所有参数生效。这样,用户就完成了 WAP 手机的设置,此时在主屏幕中选择“浏览器”一项,即可开始访问 WAP 站点。

3.2.4 西门子 3568i 手机上网设置

西门子 3568i 手机上网的主要设置如下:

(1) 在手机待机状态,在主屏幕中选择打开“菜单”,并从中选择“工作娱乐”命令。

(2) 再从出现的项目中选择“互联网”,在打开的列表中选择“设定配置文件”,进入设置配置文件的界面。

(3) 从中选择“配置-1”一项,然后选择“设定”,即可在出现的界面中设置各个项目。比如,在“配置文件名称”一项后,可输入一个自己喜欢的名字,比如“WAP 世界”。然后,在其他项目后,设置相应的内容,具体可参见表 3.4 所示。

表3.4 西门子3568i手机的上网设置项目及内容

设置项目	设置内容
主页地址	即设置欲浏览的 WAP 网址,如 wap.chnmobile.com、wap.wapfrum.com、wap.bookingall.com 等
拨叫号码	中国电信为“172”,中国联通为“1601”
网关地址	中国电信为“10.0.0.172”,中国联通为“192.168.167.2”
持续时间	“300”秒
数据通话类型	“模拟”
数据通话速度	“9600”b/s
登录姓名	“wap”(小写)
密码	“wap”(小写)
线路类型	“调制解调器”

需要说明的是,输入过程中,连续按 4 次“0”键可以输入一个句点(.),连续按 2 次“*”键可以输入一个斜杠(/)。其他符号的输入方法,具体可参见手机使用说明书。

(4) 设置完后返回“WAP 世界”菜单，从中选择“激活”命令，即可使新设置的参数生效。以后，用户就可以通过手机访问 WAP 站点了。

本章小结

本章介绍了 WAP 手机上网前的准备工作，并以诺基亚 7110、爱立信 R320sc、摩托罗拉 V8088/L2000www/LF200www/T2288、摩托罗拉 A6188、西门子 3568i 等 WAP 手机为例，讲解了手机上网的设置项目、设置内容及具体的设置方法。

本章内容非常实用，不仅可以为那些使用 WAP 手机上网的用户提供设置操作方法，而且可以为那些利用 WAP 手机进行实际开发测试的开发人员提供设置操作参考。

第 4 章 WAP 网站的服务器建设

第 4 章 WAP 网站的服务器建设

为了测试我们开发的 WAP 网页及应用, 我们需要建立 WAP 服务器环境。前面曾经讲过, WAP 服务器可以在已有的 WWW 服务器上通过建立 WAP 站点的形式来建立。而 WWW 服务器可以使用 IIS 4.0/5.0 或 PWS 等系统来建立, 也可以使用专门的 WAP 服务器软件如 Nokia WAP Server 1.1 等来建立。一般来说, 专门的 WAP 服务器软件的规模比较小, 安装和设置方法比较简单, 我们就不详细介绍了。我们以比较常用的 Windows NT/2000 下的 IIS 4.0/5.0 和 PWS 为例, 介绍 Web 服务器的建立和配置方法, 以及在 Web 服务器上构建和配置 WAP 服务器的具体方法。

4.1 Web 服务器构建概述

Web 服务器的构建需要两个必不可少的基础平台, 即网络硬件平台和网络软件平台。只有完成这两个平台的建设, 才可以建设 Web 服务器。

(1) 网络硬件平台的搭建。通常采用局域互联技术, 建设 Web 服务器所需的局域网, 然后再将局域网与 Internet 相连, 从而为实现 Web 服务器与 Internet 相连提供硬件基础。目前, 局域网方面的主流技术有以太网、FDDI、ATM 等, 其中最常用的是以太网, 它具有价格低廉、实施方便等特点。有关这方面的内容涉及较多的局域网建设知识, 如有需要, 请大家参阅相应的书籍。

(2) 网络软件平台的搭建。由于有线网络中的网页信息均通过 HTML 格式进行 Web 发布, 无线网络中的网页信息均采用 WML 或 HDML 格式进行 WAP 发布, 所以欲建的 Web/WAP 软件平台必须以 TCP/IP 协议为基础, 并提供和支持 HTTP 传输协议。WAP 平台还需要支持 WAP 协议。目前比较成熟的网络操作系统有 UNIX、Novell Netware 和 Windows NT/2000 等, 其中 UNIX 功能强大但操作十分复杂, 维护也不方便; Novell Netware 功能也

不错,但它的用户界面不甚友好;Windows NT/2000 具有 Windows 95/98 一样的用户界面和交互形式,而且维护起来比较方便。它们都可以用做 Web 服务器的软件平台,但考虑到实用性,我们通常选择使用易于管理和易于维护的操作系统,如 Windows NT Server 4.0 或它的升级版本 Windows 2000。

考虑到中国市场的巨大需求,Microsoft 公司还针对中、小型企业环境特别推出了 Small Business Server 4.0 中文版。这是一个基于 NT 环境的集成软件包,包括 Windows NT Server、Microsoft Exchange Server、Microsoft SQL Server、FrontPage、Proxy Server、Modem Sharing Server 等。使用这套软件建立网站的软件平台时,不仅可以为企业节省大量的费用和管理时间,同时也可以为企业丰富而实用的功能。

一般来说,以 Microsoft 的 Windows NT 操作系统建立网络平台后,再在这个平台上搭建 Web 服务器就比较容易了。Microsoft 公司免费提供的 Internet Information Server(IIS)系统可以建立 WWW 和 FTP 等服务器,提供基于 Web 的运行环境。而且,IIS 建立的服务器能够直接利用 NT 特性,如安全性、多线程等,能够与 NT 的内核紧密集成,从而提供比较高的处理效率。由于 IIS 是目前比较常用的建立 Web 服务器的信息服务系统,本章我们就以它的最新版本 5.0 为例说明 Web 服务器的安装与设置方法。目前,更为常用的 IIS 版本是 4.0。IIS 5.0 与以前的版本相比有了较大的变化,增加了一个 E-mail 服务器功能,同时将原来的 Gopher 服务器功能去掉,这种服务器现在已经极少有人使用了。不过,我们在发布网页中需要的 Web 服务器属于 WWW 服务器,这种服务器在 IIS 5.0 和 4.0 中都是一致的,建立方法完全相同,所以我们介绍的方法也完全适合于 IIS 4.0 的用户。

关于 Web 服务器的建立,我们还有两点需要特别说明:

(1) 除 IIS 外,可建立 Web 服务器的系统还有很多,如 Netscape Netsite Server、Super Web Server、W4Server、Website、WebQuest、NCSA HTTPD Server 等。用户也可以选择使用其他的系统来建立 Web 服务器。

(2) 对许多学习 WAP 的普通用户来说,可能不具备局域网环境,不能使用 IIS 来建立 Web 服务器,进而再建立 WAP 服务器。但可以使用 Microsoft 公司专门为这种情况提供的个人 Web 服务器系统 PWS(Personal Web Server),它是一个桌面 Web 服务器,使用它用户可以从自己的计算机上建立站点服务器并发布个人主页。PWS 尤其适合作为一个网站建设的开发平台,它使得用户在同一台计算机上就可以实现网页发布、站点测试等的模拟网络功能。PWS 的安装与 IIS 的安装类似,配置操作更为简单,我们本章也简单介绍一下。

4.2 IIS 的安装基本配置

Internet 信息服务器 IIS(Internet Information Server)是一个用于出版 Web 内容和 FTP 的可缩放式的企业网工具,可以建立包括 WWW 和 FTP 在内的信息服务器。随着 Internet 和 Intranet 站点的不断增加和广泛普及,IIS 的用途变得越来越重要。下面,我们在简单分析 IIS 响应客户请求方法的基础上,讲解 IIS 的安装及 WWW 服务器的配置操作。

4.2.1 IIS 响应客户请求的方法

作为 Web 服务器,IIS 需要完成响应客户端用户发来的请求。那么怎么响应呢?我们先分析客户机 Web 浏览器与 Internet 上 Web 服务器之间进行连接和通信的过程,然后再说明 IIS 响应客户请求的方法。

我们知道,基于 Internet 的通信以及 IIS 都建立于 TCP/IP 协议的基础上。TCP/IP 有几个核心协议,正是由于它们的相互作用或互相补充,才有效地完成了网络中的通信。表 4.1 给出了这些协议的功能解释。了解了这些协议的功能,我们就可以讨论客户机与 Internet 远程主机之间进行连接和通信的过程了。如图 4.1 所示,这一连接和通信过程主要完成以下几步内容:

(1) 用户在运行 Web 浏览器的客户计算机中,输入一个想要访问的 URL 地址,比如人民日报的网址 <http://www.peopledaily.com.cn>,然后按回车键确认该请求。

表4.1 TCP/IP的几个核心协议

协议	功 能
ARP	地址解析协议 ARP(Address Resolution Protocol)用于按照已知的 IP 地址来查找计算机的硬件地址,它是当客户机与远程主机建立连接时,首批执行的协议之一。
TCP	传输控制协议 TCP(Transmission Control Protocol)是用于实际传送数据的协议。当客户机向远程主机请求一个 Web 文档时,TCP 协议就负责传送其中的数据信息。
IP	互联网协议 IP(Internet Protocol)用于寻址、路由以及转发网络包。当客户机指定了要到达的 Web 站点时,该请求的具体路由就由 IP 协议确定。
ICMP	互联网控制协议 ICMP(Internet Control Message Protocol)是一个错误和状态报告协议。当客户机的请求无法到达目的地时,ICMP 就使用一条“目标主机不可到达(Destination Host Unreachable)”的消息作出响应。
IGMP	互联网管理协议 IGMP(Internet Group Management Protocol)主要负责组注册。客户机一旦启动,IGMP 就马上向路由器通告这台客户机已经出现在网络上,并将该客户机列在路由器的清单中。

续表

协议	功 能
UDP	用户数据包协议 UDP(User Datagram Protocol)也是一个传输协议。如果用户启动的客户机是一台基于动态主机配置协议 DHCP(Dynamic Host Configuration Protocol)的机器,则该机器一旦启动就会向 DHCP 服务器申请一个 IP 地址,而这个请求就是由 UDP 协议负责发送的。
FTP	文件传输协议(File Transfer Protocol)不仅是 TCP/IP 的核心协议,而且是一个应用程序。使用 FTP 协议可以把文件传输给 TCP/IP 主机,或传输来自于 TCP/IP 主机的文件。
HTTP	超文本传输协议 HTTP(HyperText Transfer Protocol)是用于客户机请求 Web 文档的协议,与 IIS 密切相关。

(2) 客户计算机向 Internet 上一台已配置好域名系统 DNS(Domain Name System)的服务器发出请求。

(3) DNS 服务器首先检查自己的数据库记录,看能否解析出该 URL 的 IP 地址。如果不能,则继续查询其他的 DNS 服务器,直至查找到可以解析该 URL 的数据记录。

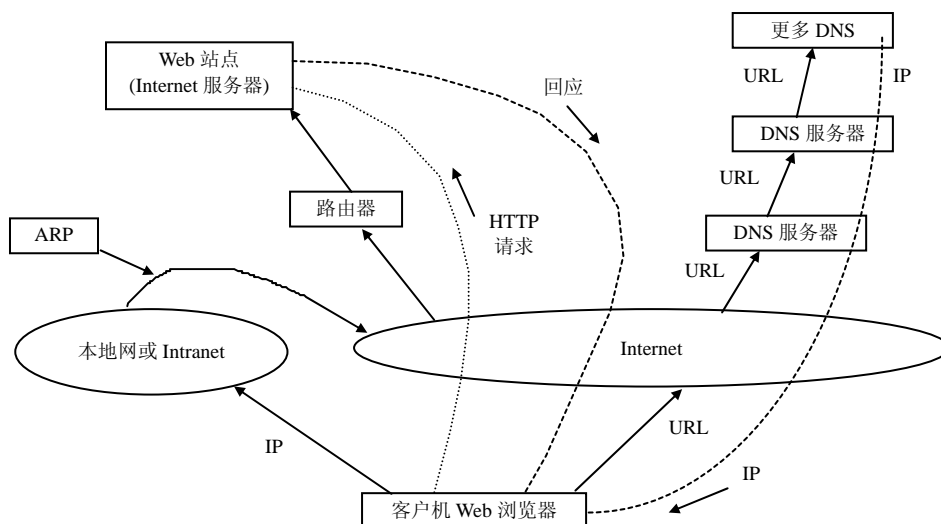


图 4.1 Web 浏览器与 Web 服务器的连接与通信

(4) DNS 服务器用解析所得的站点的 IP 地址来响应客户的 URL 解析请求,即将结果返回给客户机。

(5) 客户机收到后,首先试着在本地网中连接该 IP 地址定义的 Web 站点。如果不能连上,则接着使用 ARP 协议来寻找能够转发该网络通信请求的路由器,也就是路由器的硬件地址。

(6) 找到路由器的硬件地址之后,客户机就把一个 HTTP GET 请求发送给这一地址对应的 Web 站点。当然,客户机究竟请求哪一种文档(网页、文件还是表单等),完全取决于客



户端用户的输入或远程 Internet 服务器的配置。

(7) 远程的 Internet 服务器查看发出请求的用户是否拥有访问所请求文档的权限。如果有, 则发出用户要求的文档。否则, 返回拒绝信息。

一般来说, Web 站点的每个组成成分, 如图形、图框、文字等, 都必须通过单独的 HTTP GET 请求和建立文件会话来进行访问。所以, 当 Web 浏览器与 Web 服务器建立连接和通信时, 要频繁执行上述过程。每当客户机的用户从其当前正在连接着的域外部请求某一资源时, 就会频繁使用 ARP 协议和 DNS 服务器, 并不断寻找合适的 Web 服务器, 直到找到需要的 Web 服务器时为止。

显然, 这一过程中远程的 Internet 服务器可能会给客户发回许多种响应。这一任务主要有 IIS 负责完成。发回响应时所采用的通信协议有多种, 如 HTTP 或 FTP 等。FTP 用于传输文件, IIS 可以专门提供相应的途径预以实现。HTTP 用于传输各种文档及完成 Web 浏览器与 Web 服务器之间的绝大多数通信。一般而言, 通过 HTTP 协议返回文档时, IIS 通常使用静态文件、CGI 脚本、ISAPI 扩展程序等几种方法中的一种来响应客户请求。

4.2.2 IIS 的支持服务

目前 IIS 的最高版本是 IIS 5.0, 它除含有用于为 Web 浏览器准备 HTML 文件的核心 IIS 部件外, Microsoft 还为加强 IIS 功能而提供了一些支持性的服务, 主要包括下述 5 种应用服务, 其中与我们建立 WAP 网站服务器密切相关的是前两种。

(1) 微软管理控制台 MMC。IIS 的所有设置都是通过 Microsoft Management Console 来控制 and 管理的。它的界面类似于“Windows 资源管理器”的界面, 整个窗口被分为两个窗格。左侧窗格用于显示网络的层次结构, 属于范围窗格; 右侧窗格用于显示当前选中文件夹的内容, 属于结构窗格。MMC 使用了一种叫作插件(Snap-ins)的管理程序。从用户账号管理到配置 Windows NT 安全策略的所有场合, 都有相应的插件。当用户安装 IIS 时, MMC 的插件 Internet 服务管理器(Internet Service Manager)也同时安装到系统中, 并与 MMC 一起管理和控制 IIS 及其相关组件。

(2) 微软事务服务器 MTS。Web 站点的主要用途是提供对后台数据环境的实时访问。如果 Web 站点只提供数据的简单显示, 那么通过使用 HTML 页面和 ASP 或 CGI 脚本格式的动态内容, 就可以执行和完成必要的访问内容处理和管理功能。但是, 现在的 Web 环境



变得日益复杂,很多情况下 Web 服务器需要执行十分复杂的互动操作,比如处理客户定单、库存清单或应付账款等,此时仅靠 HTML、ASP 或 CGI 就不能胜任这一工作了。为此,Microsoft 提供了集成软件包 Microsoft Transaction Server,能够提供把多个操作当作单个操作来执行的能力。这一事务支持是 Internet 上成功实现金融活动的关键。例如,现在有一用户使用 Internet 把资金从一个账号中调出来,然后把这笔资金转存到另一账号中。此时,如果这笔资金从原来账号调出之后,但在转入另一账号之前,系统出现了故障,那么用户就要失去这笔资金。——显然,这一情况是不允许发生的,否则基于 Internet 的电子商务就根本无法实现。而可喜的是,MTS 可以保证整个转账操作过程的安全和连续。只要出现任何没有得到成功执行的环节和步骤,它就会撤销整个操作,让用户重新操作,以使用户资金免受损失。

(3) 认证服务器 CS。由于越来越多的企业、公司通过 Internet 开展电子商务的业务活动,所以安全可靠的数据传输变得越来越重要。为了保护数据在 Web 浏览器与 Web 服务器之间的安全交换,人们建立了安全会话层 SSL(Secure Socket Layer)协议及其他底层安全协议负责其间的数据交换,同时,还需要一个第三方的认证管理机构对参与数据交换的双方进行身份或信誉认证。Certificate Server 就是用来生成 SSL 密钥和对数据进行加密、解密的服务器应用程序,可以建立 Internet 上的认证管理机构,实现数字认证的发布、延期和解除操作。使用这些数字认证,能使 Internet 上的用户通过线路发送个人信息(如信用卡号和密码等)而无需担心数据被人截获或窃听。

(4) 索引服务器 IS。Index Server 用于提供 Web 站点的搜索引擎功能。它可对 Web 服务器上的所有 Web 页面和含有扫描过滤器的其他文档进行扫描,并建立相应的索引,这一索引中含有它扫描发现的每个单词。当用户通过 Web 浏览器进行搜索时,Web 服务器就会把含有搜索短语的所有站点数据或页面数据通过一个 Web 文档来返回给用户。用户可以通过单击其中的超链接(搜索结果)来查看搜索到的相应文档。

(5) 活动服务器页面 ASP。借助 Internet 来处理电子商务业务或完成数据、信息访问基本上都是以访问数据库中的信息为基础,而 Active Server Pages 就可以在 Web 页面需要的时候动态地生成所需的 Web 页面,并提供 IIS 与数据库连接的功能。ASP 是建立动态 HTML 文档的一种方法。所谓动态文档,是指文档中的数据在用户每次把文档装入到 Web 浏览器中时都是可变的,也就是动态的。ASP 通过使用嵌套在 HTML 文档中的服务器端脚本描述来实现数据的变化,也就是说脚本运行在进程的服务器端上,而不运行在客户端上。其好



处是，客户端的浏览器无需理解脚本描述语言就能运行含有 ASP 的页面。不过，服务器端为了使用 ASP，还必须拥有含有数据的数据库，而这又必须依赖其他相关的服务，比如微软数据访问组件 MSDAC(Microsoft Data Access Components)等。

4.2.3 IIS 的服务账号

作为网络操作系统，Windows NT Server 禁止对系统内资源的任何未经授权的访问。也就是说，对任何登录并试图访问服务器资源的客户，NT 都会对照系统的资源权限列表来核查用户的登录，以验证每个客户资源访问请求的合法性。IIS 安装于 NT 系统之上，它所出版的所有 Web 站点及 Web 文档内容都存放于安装 NT 系统的服务器中，所以通过 Internet 访问 Web 内容即服务器资源的远程用户也必须通过 NT 的合法性检验。不论该 Web 内容属于 Internet 还是属于 Intranet，只要是由 NT 系统来管理，那么如果没有符合要求的权限，远程用户就无法访问这些 Web 文件。即便是本地网络用户请求访问服务器中的数据库，或者请求打印其中某个文档时，NT 也会检验该用户请求的合法性。

例如，如果远程 Internet 用户在其 Web 浏览器中输入了想要访问的一个 URL 地址(如 <http://www.online.sh.cn>)，那么他的账号就必须拥有访问该站点的权限。但是，包含它所访问站点内容的 Windows NT Server 可能是他第一次访问，那么他的用户账号怎么才能拥有访问权限呢？这就是 IIS 服务账号发挥重要作用的地方。

世界各地登录 Internet 访问的用户有很多，是不是需要在每台服务器的 NT 系统中为所有可能访问的用户都建立账号呢？这显然是不可能的，也没有这种必要。Microsoft 的解决办法是使用同一个 IIS 服务账号去替代每个可能访问用户的账号来进入 NT 系统。这样，所有用户都可以作为匿名用户访问 IIS 和其中的 Web 站点。

安装 IIS 时，安装程序会自动创建一个 IIS 服务账号。缺省状态下，这一账号被命名为“IUSR_服务器名”形式的账号，其中“服务器名”是安装 IIS 的服务器的名称。IIS 建立这一账号之后，自动给这个账号赋予“域用户”和“客人”组成员的关系。IIS 同时还会设置账号属性，使 Internet 用户无法改变口令，并让口令永不过期。IIS 之所以不允许用户更改账号口令，是因为该账号是一个为许多用户所共用的公共账号；不允许账号口令过期，是因为考虑到这是 Internet 用户访问 IIS 服务器时所需的账号，IIS 服务器必须提供长期稳定的账号，以长期供世界各地的用户访问使用服务器资源。




4.2.4 IIS 的安装

对 IIS 及其相关组件有了上面的认识后,我们再介绍最为实际的内容,也就是安装 IIS 系统,以便为以后建设 Web 站点等提供信息服务平台。

目前 IIS 的最高版本是 IIS 5.0,随 Windows 2000 一起推出,在安装 Windows 2000 时它会自动安装。所以有关 IIS 5.0 的安装我们就不多解释了。因为当前比较常用的版本是 IIS 4.0,它属于 Windows NT 4 Option Pack 的一部分,需要单独安装。我们这里着重介绍一下 IIS 4.0 的安装。

(1) 在服务器计算机开启状态,把 Windows NT 4 Option Pack 光盘放入欲安装 IIS 服务器的 CD-ROM 驱动器中,关闭驱动器,Windows NT 4 随后会自动读取光盘中的内容,启动安装 Windows NT 4 Option Pack 的向导界面,如图 4.2 所示。

(2) 从中单击“下一步”按钮,安装向导首先给出“最终用户许可协议”,用户必须接受该协议并单击“接受”按钮,才能继续下面的安装。

以后操作中,完成每一对话框的选择与设置后,单击“下一步”按钮可以继续安装,单击“上一步”按钮可以返回修改。

(3) 接下来安装向导让用户选择想要执行的安装类型。可以选择“升级”或“升级之外”。前者只用于升级当前 NT 系统,后者除升级之外还安装 IIS 的相关组件。我们这里选择“升级之外”,然后单击“下一步”按钮,安装向导进入“组件”对话框,显示有 IIS 的各个组件,用户需要从中选择想要安装各个组件,然后单击“下一步”按钮。

(4) 接下来,安装向导让用户指定 IIS 在硬盘上的安装位置,如图 4.3 所示。一般选择默认值。

(5) 单击“下一步”按钮,安装向导显示对话框,让用户指定“事务服务器(Transaction Server)”的安装位置,如图 4.4 所示。我们这里也是选择默认值。

(6) 随后,安装向导显示“事务服务器管理账号”对话框,如图 4.5 所示。它让用户选择账号类型。这里采用默认设置即可。

(7) 单击“下一步”按钮,安装程序将进行文件复制和安装,并显示安装进程。最后就可以把 IIS 4.0 及其组件安装到 NT 系统中。

安装完毕后,NT 系统“程序”菜单中将出现“Windows NT 4 Option Pack”子菜单,

其中含有 IIS 及相关组件的菜单命令或更下一级的子菜单,如图 4.6 所示。利用其中的命令,用户就可以配置 IIS 服务器并建设、出版 Web 站点等信息服务了。

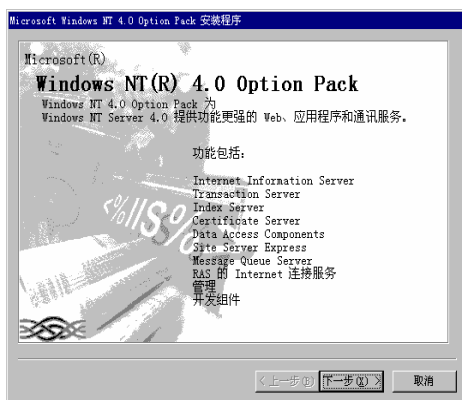


图 4.2 安装初始界面



图 4.3 选择 IIS 安装位置



图 4.4 选择 MTS 的安装位置

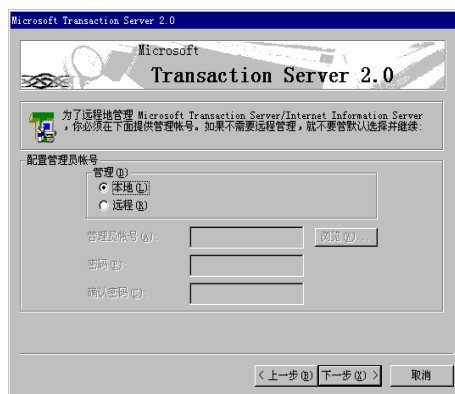


图 4.5 选择 MTS 账号

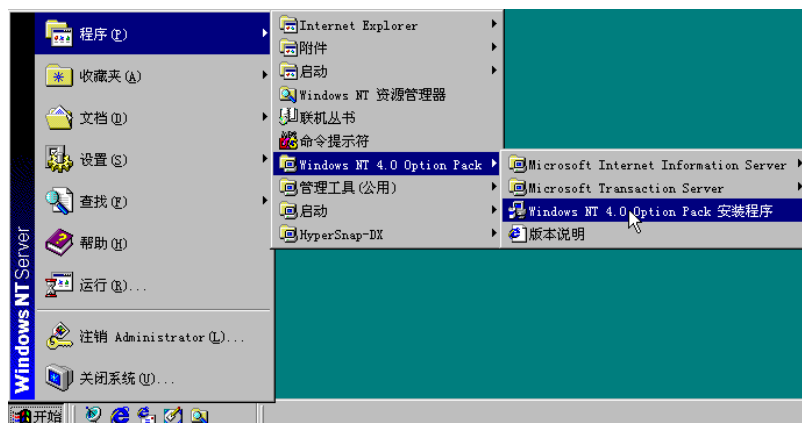


图 4.6 “Windows NT 4 Option Pack”子菜单及其命令

4.2.5 IIS 的基本配置

为顺利使用 IIS，用户还需要对 IIS 进行有关参数与项目的设置、配置。具体操作时可通过 Microsoft 管理控制台 MMC(Microsoft Management Console)来实现。

(1) 在 NT 系统中单击“开始”按钮，打开它的菜单，依次将鼠标指向“程序”→“Windows NT 4 Option Pack”→“Microsoft Internet Information Server”→“Internet 服务管理器”，即可启动 MMC，进入它的操作窗口，如图 4.7 所示。

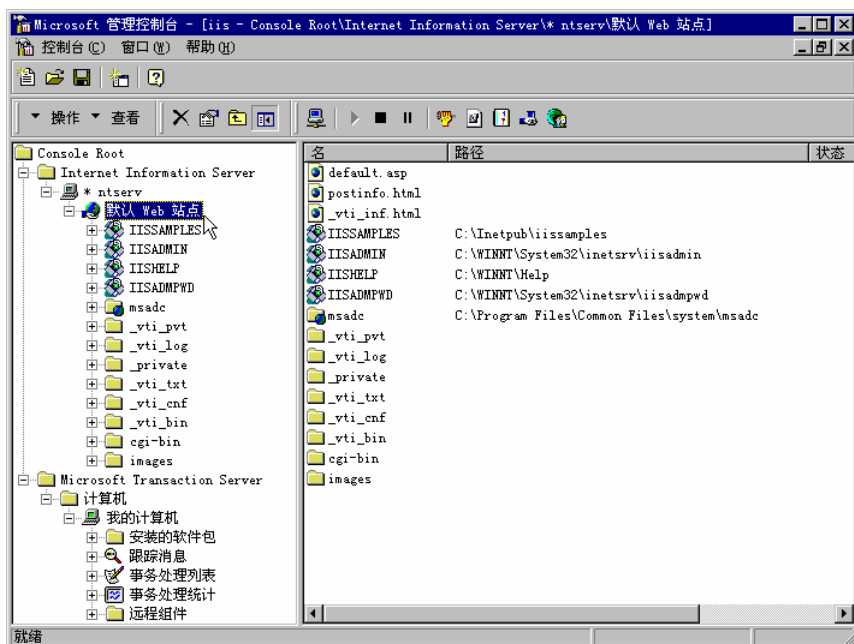



图 4.7 IIS 的管理界面——MMC

 **注意**，如果是在 Windows 2000 系统中，则需在“程序”菜单中选择“管理工具”子菜单中的“Internet 服务管理器”命令，才可打开 MMC 的操作窗口。

(2) 从 MMC 左侧窗格中展开 Internet Information Server 组。MMC 将显示 IIS 服务器。

(3) 选中该 IIS 服务器并单击鼠标右键，从出现的快捷菜单中选择“属性”命令，屏幕上将随后出现当前服务器的属性对话框，如图 4.8 所示。

(4) 单击该对话框中的“编辑”按钮，可打开“默认 Web 站点属性”对话框，如图 4.9 所示。这一对话框中共有 9 个选项卡，通过这些选项卡中的项目即可对 IIS 服务器进行设置。各选项卡的主要功能介绍如下：

- “Web 站点”。该选项卡含有站点的配置信息，以及服务器可以处理的最大并行连接个数、日志文件有关项目等。
- “性能”。该选项卡用于调整 IIS 服务器的性能。使用其中的滑动条，可以调整 IIS 支持的用户数量，也可以调节带宽流量等。



图 4.8 服务器属性对话框

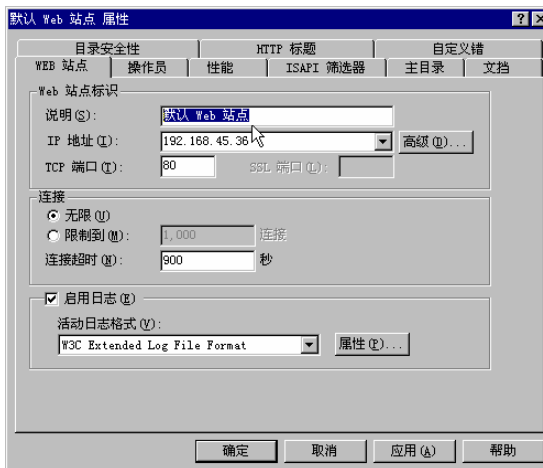
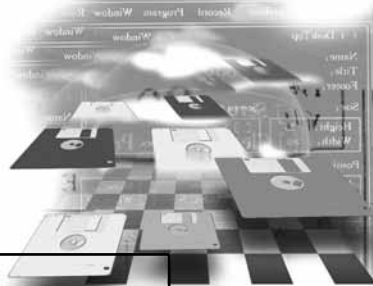



图 4.9 “默认 Web 站点属性”对话框

- “操作员”。该选项卡用于完成现有 NT 域账号的操作员授权及其权限管理。
- “主目录”。使用该选项卡可以指定主目录的位置，是在本地服务器，还是在远程服务器。而且还可以设定主目录的访问权限。
- “ISAPI 筛选器”。这一选项卡给出了 IIS 服务器上安装的过滤器的选择列表。这里的过滤器都是 IIS 来处理信息请求的 DLL 文件，用户可以根据需要选择使用。
- “自定义错”。当试图访问某一 Internet 站点时，用户可能会收到一条“文档不存在”的错误消息。这一功能就是由该选项卡来定制的，其中一并可以指定应答客户请求的具体信息。
- “文档”。用于设定缺省文档值。所谓缺省文档是这么一回事，Internet 用户访问一台 Web 服务器时，如果没有指定具体文档，如 www.peopledaily.com.cn/pcar.html，那么该服务器就会使用“文档”选项卡中定义的缺省文档来作出响应。这一文档通常是本站点的主页。
- “HTTP 标题”。该选项卡用于配置 HTTP 网页的头部内容，主要是配置内容等级、内容有效期和文件类型。内容等级是为限制色情、暴力、恐怖方面的网页而新增的过滤功能。



- “目录安全性”。IIS 管理员使用这一选项卡可以设置准许访问 IIS 的安全权限；以及配置 SSL(Secure Socket Layer)通信，以便允许或禁止匿名访问。而且，还可以设定是根据 IP 地址还是 Internet 域名来否决对请求站点的访问。

 我们这里只是给出了 IIS 安装与设置的基本步骤或内容，更详细的操作方法由于需要较多的叙述篇幅，我们这里就不展开了，感兴趣的读者请参考有关书籍。

4.3 WWW 服务器的建设管理

完成 IIS 服务器的安装和配置之后，我们就可以建立用于发布 Web 站点的 WWW 服务器了。WWW 服务器建设的核心是建立 Web 站点，包括 Web 站点设置、主页文件及目录设置、目录安全设置、错误信息设置等。

4.3.1 创建新的 Web 站点

事实上，一个 Web 站点仅仅是 IIS 服务器上的一个目录，而且该目录的访问权限由 IIS 负责控制。当 Internet 上的用户遍历 Web 站点时，他们实际上是在查看远程服务器上一个一个目录(或虚拟目录)的内容。建设 Web 站点的过程也就是在 IIS 服务器上建立目录，并把 IIS 指向这一目录，且使该目录能让 Internet 用户进行访问的过程。

正确安装好 IIS 后，它将在系统中自动建立一个称为“默认 Web 站点”的站点。用户可以直接使用这个站点发布网页，也可另建其他站点使用。

建设 Web 站点前，首先要在 IIS 服务器上规划好站点目录结构，并建立这些目录，同时记下各个目录名。然后，运行 IIS 的组件 MMC 即可完成站点建设。具体步骤如下：

- (1) 采用前面介绍的方法(参见 4.2.4 节)，启动 MMC，进入它的操作窗口(图 4.7)。
- (2) 从 MMC 左侧窗格中展开 Internet Information Server 组。MMC 将显示 IIS 服务器。
- (3) 选中该 IIS 服务器并单击鼠标右键，从出现的快捷菜单中选择“新增”子菜单中的“Web 站点”命令(图 4.10)，IIS 即会启动“新 Web 站点向导”，并在屏幕上打开它的对话框，如图 4.11 所示。
- (4) 在该对话框中，为即将要建立的站点输入一个说明，比如“MyWorks”，随后单击“下一步”按钮，进入新 Web 站点向导的下一对话框，如图 4.12 所示。

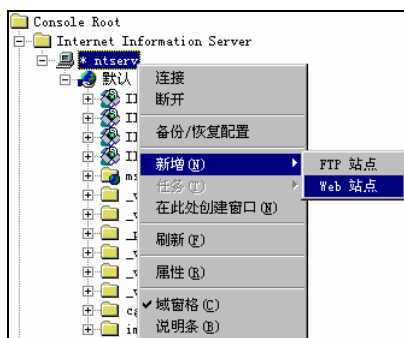


图 4.10 选择“Web 站点”命令



图 4.11 输入 Web 站点名称

(5) 单击“选择此 Web 站点使用的 IP 地址”下拉列表，从中选取可以使用的 IP 地址。此外，在这一对话框中还可设置想要使用的 TCP 端口号，如 80。然后单击“下一步”按钮继续。

(6) 这时出现又一对话框，如图 4.13 所示。该对话框让建站人员把含有 Web 内容文档的目录路径输入到“输入你的主目录路径”字段框中。如果希望所有 Internet 用户都能访问这一站点，则要确保选中“允许匿名访问此 Web 站点”复选框。然后单击“下一步”按钮，进入新 Web 站点向导的下一对话框，如图 4.14 所示。



图 4.12 选择 IP 地址和端口

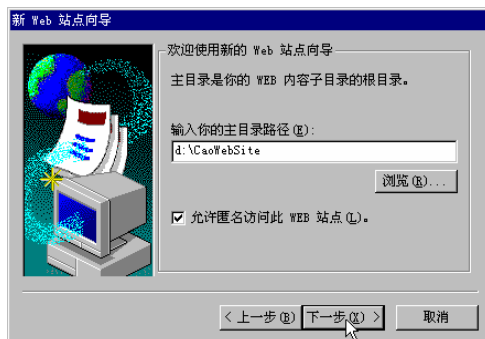


图 4.13 输入目录路径

(7) 这是最后一步设置内容。在该对话框中，建站人员需要为当前建立的站点选择访问权限。如果希望使用一个访问计数器来累计访问人数，则必须允许“执行”和“写”权限。最后单击“完成”按钮，即完成 Web 站点的建立。

以后，Internet 用户就可以使用建站人员在出版期间所指定的 IP 地址本访问这一 Web 站点。不过，为了能够让 Internet 用户使用主机和域名，如 www.mycweb.com 等来访问站点，则还必须在 DNS 服务器上建立一条记录，以便将域名解析为 IP 地址。

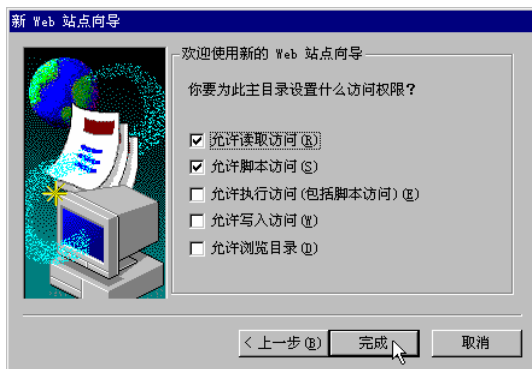


图 4.14 设置访问权限

4.3.2 配置 Web 站点

站点建立后，用户还可以根据具体需要配置 Web 站点。操作方法如下：

- (1) 启动 MMC，进入它的操作窗口(图 4.7)。
- (2) 从 MMC 左侧窗格中展开 Internet Information Server 组。MMC 将显示 IIS 服务器中的 Web 站点列表。
- (3) 选中想要配置的 Web 站点并单击鼠标右键，然后从出现的快捷菜单中选择“属性”命令，即可打开当前所选站点的属性对话框。图 4.15 所示就是我们打开的默认 Web 站点的属性对话框(同图 4.9)，当前处于打开状态的是“Web 站点”选项卡。

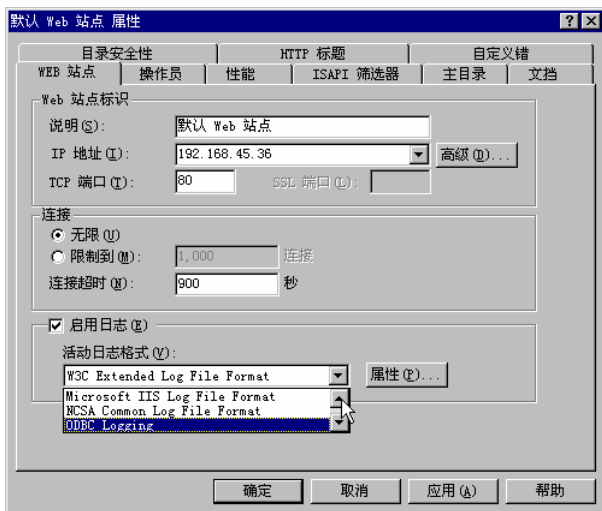


图 4.15 “Web 站点”选项卡

- (4) 在该选项卡中，共有 3 个区域的若干项目需要设置，下面我们就分别介绍一下。

① “Web 站点标识”区域中共有 4 项需要设置：

- “说明”。指对当前站点服务器的描述，也就是站点名称，它是用来识别服务器站点的控制名的。在 NT 系统中，IIS 可以安装最多 16 个 Web 服务器。当一台计算机中同时装了多个服务器时，每个服务器都应当取一个可以识别的名字。常用的取名方法通常有两种：一种是直接使用主机名或任意起一个名字，另一种是使用计算机的域名地址。我们这里采用默认名称，即“默认 Web 站点”。

- “IP 地址”。这是一个下拉列表，从中可以为当前站点的服务器选择 IP 地址。对于一台主机来说，它的域名地址可以是任意的，也可以同时拥有多个域名地址，但它的 IP 地址只有一个。设置 IP 地址之前，必须先向主管机构申请一个 IP 地址。如果用户所用网络是内部局域网，仅供开发测试之用，这时可随意指定一个 IP 地址。单击旁边的“高级”按钮，可以打开“高级 Web 站点配置”对话框，如图 4.16 所示。其中列出了当前站点的不同标识，单击其中的“添加”按钮，可以打开图 4.17 所示的“高级 Web 站点标识”对话框，从中可以选择或指定 IP 地址、TCP 端口及主机标识名。



需要注意的是，设置 IP 地址时千万不能出错，否则服务器将不能正常工作。

- “TCP 端口”。用于设置 TCP 端口号，它是计算机连到 Internet 上的出入口。每一个服务器都要有一个端口号，而且不同服务器的端口号应当设置不同。理论上，端口号可以在 1 到 65 535 中选择。不过，实践中人们已经形成一些约定性的基本的端口设置，如 HTTP 服务器的端口号为 80、SMTP 服务器的端口号为 25、Telnet 服务器的端口号为 23、FTP 服务器的端口号为 21、HTTPD 服务器的端口号为 443 等。如果将 HTTP 服务器的端口号设为 80，则访问时可直接使用域名地址或 IP 地址，如 <http://www.mywebs.com> 或 202.111.112.113 等。否则，访问主机时就必须指明端口号，如 <http://www.mywebs.com:8080> 等。8080 号端口通常用来作为代理服务器的 TCP 端口号。

- “SSL 端口”。用于设置使用 SSL 协议时的端口号。

② “连接”区域共有 2 项需要选择及设置：

- “无限制”及“限制到”单选项。通过这两个单选项用户可以设置同一时刻外面的用户与主机连接的数目：无限制或限制具体的数目。一般地，同一服务器可以同时响应外部用户的多个连接请求。对于每一个用户请求，服务器都会自动启动一个进程来响应该请求，而这一进程在等待状态下将占用系统的大约 200KB 的内存，在启动后将占用大约

300~500KB 的内存，所以设置时，一般要视具体的情况而设置限制连接数，不宜设置过大，否则会影响系统速度和资源利用。通常不选择“无限制”，默认的限制连接次数为 1000 次。



图 4.16 “高级 Web 站点配置”对话框



图 4.17 “高级 Web 站点标识”对话框

- “连接超时”。用于设置连接等待的时间，单位为秒。当外部用户向主机发出连接请求时，主机会相应地启动一个进程来处理这一请求。如果主机与外部用户建立了连接，则服务器进程将处于等待用户的数据请求的状态中。如果外界用户经过一段时间还没有发出数据请求，则主要需要考虑断开这一连接，目的是避免系统资源浪费，提高系统效率。主机在断开此次连接前需要等待的时间，就是由本项所定义。具体设置时应根据实际情况决定。如果服务器的传输速率比较慢，或者连接到服务器上的外部用户比较少，则可以将时间设置得长一点，否则就设置得短一些。默认的超时设置为 900 秒。

③ “启用日志”区域。“启用日志”本身就是一个复选项，选择该项后，才可以从其下面的“活动日志格式”下拉列表(见图 4.15)中选择需要的日志格式。

日志文件是保障服务器安全的一个十分重要的工具，其主要作用是监视外界用户与当前主机的连接信息，包括外界用户的主机名、连接时间、连接活动等信息。管理员通过系统日志可以查看到那些访问服务器的外部主机、外界用户的 IP 地址等重要信息，从而为以后进行安全方面的设置提供决策参考。

可以选择使用的日志文件格式有 3 种：第一种为 Microsoft 的 IIS 系统日志格式；第二种为 NCSA 的公共日志格式，也就是 HTTPD 日志格式；第三种为 W3C 扩展日志文件格式。

由于日志文件不断地记录系统活动，所以日志文件会不断地增大。为防止文件过大，用户必须定期进行更新。更新之前，一般需要将文件保存起来，以备后用。而且，用户可以设置日志文件的更新周期。操作方法十分简单：在“活动日志格式”下拉列表中选择需

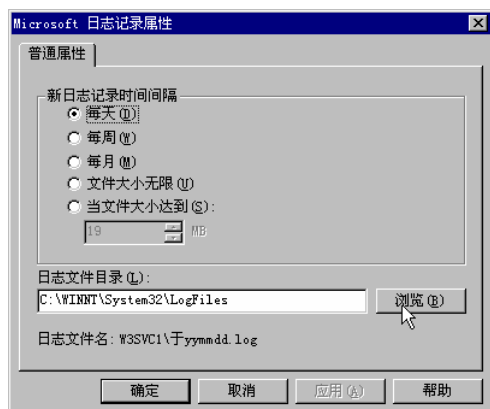


图 4.18 设置日志文件属性

要的格式类型，比如选择 Microsoft 的 IIS 日志格式，然后单击它旁边的“属性”按钮，打开它的对话框，如图 4.18 所示，从中即可设置日志记录时间间隔。另外，从中还可以选择或指定“日志文件目录”，操作都比较简单，我们就不一一展开了。

(5) 完成上述各项设置后，返回到图 4.15 所示的“Web 站点”选项卡，单击“确定”按钮，即可完成 Web 站点的配置工作。

4.3.3 配置主目录

前面讲到，建立 Web 站点时要指定存放 Web 文档的主目录，远程用户访问的缺省文档就在这一主目录中。主目录和主页文件名在安装服务器时都是可以设置的，设置主目录的目的是告诉浏览器到什么地方去找网页文件。例如，在站点 <http://www.mywebsite.com> 的主目录下放了一个名为 `hello.htm` 的网页文件，则在浏览器中输入“<http://www.mywebsite.com/hello.htm>”，就可以访问到该文件。

主目录并不是一成不变的，必要的时候，我们还可以把其他目录设置为主目录，以供远程用户访问。配置其他目录为主目录的操作方法如下：

- (1) 使用“Internet 服务管理器”命令启动 MMC，进入它的操作窗口(参见图 4.7)。
- (2) 从 MMC 左侧窗格中展开 Internet Information Server 组。MMC 将显示 IIS 服务器。
- (3) 在 IIS 组中，使用鼠标右键单击想要配置其主目录的站点，这会打开一个快捷菜单，从中单击“属性”命令，打开它的属性对话框，它由几个选项卡组成(参见图 4.15)。
- (4) 从中单击并打开“主目录”选项卡，如图 4.19 所示。接下来即可设置主目录。
- (5) 首先选择目标资源来源：“此计算机上的目录”、“另一计算机上的共享位置”或“重定向到 URL”。一般选择第一项。

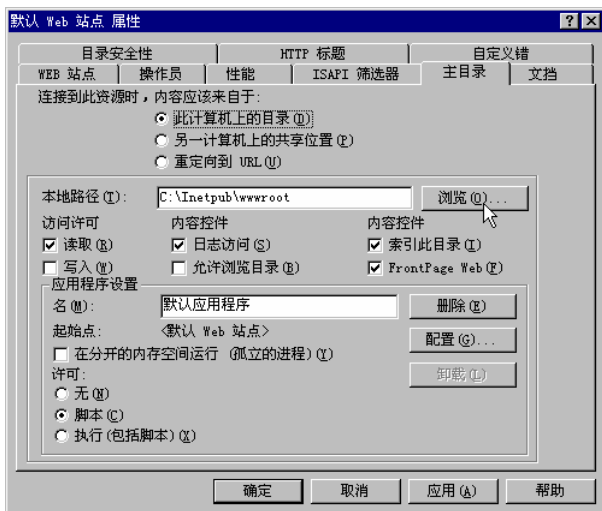


图 4.19 设置主目录

(6) 单击“本地路径”旁边的“浏览”按钮，屏幕上随后会出现选择文件夹的对话框。用户可根据需要，从中选择需要的文件夹来作为主目录，选好后单击“确定”按钮，返回“主目录”选项卡。

(7) 在“访问许可”栏下可以设置主目录的访问许可：“读取”及“写入”。在“内容控件”栏下可以设置主目录是否启用“日志访问”，是否“允许浏览目录”，是否“索引此目录”，以及是否作为“FrontPage Web”站点等。

(8) 接下来需要完成“应用程序设置”区域里的有关设置。应用程序主要包括一些访问 Web 数据库的应用程序，如 ASP 文件、IDC 文件等。这里的各项主要用于设置服务器在解释执行应用程序时所调用的相应动态链接库程序。例如，ASP 文件在解释执行应用程序时必须调用 ASP.DLL 文件，IDC 文件在解释执行时必须调用 HTTPODBC.DLL 文件。

用户从中可以指定应用程序的“名”，即名称，还可指定是否“在分开的内存空间运行”，以及指定“许可”类型等。其中最重要的是对应用程序进行配置。应用程序设置分为 3 个方面：其一，应用程序的映射设置；其二，应用程序的选项设置；其三，应用程序的调试设置。下面我们简要介绍一下这些方面的设置方法。

① 应用程序映射设置。在“主目录”选项卡中单击“配置”按钮，即可打开“应用程序配置”对话框，此时处于打开状态的是“应用程序映射”选项卡，如图 4.20 所示。

其中列出了已经建立的映射，如果用户需要增加新的程序映射，则可单击其中的“添加”按钮，打开它的对话框(图 4.21)，从中单击“浏览”按钮，打开查找文件的对话框，选择好需要的文件后单击对话框中的“确定”按钮，就可以返回到图 4.21 的对话框，同时，

所选文件的路径及文件名将出现在“可执行文件”一栏中。

而且,用户还可以指定程序所映射的“扩展名”,以及指定是否需要“脚本引擎”、“检查文件是否存在”和“不包括的方法”等。最后,单击“确定”按钮,即可返回“应用程序映射”选项卡。



图 4.20 “应用程序映射”选项卡



图 4.21 添加映射

如果发现已经添加的映射有错误或不太合适,则可选中该程序映射后,单击“编辑”按钮,打开与图 4.21 相同的对话框,从中进行修改。单击“删除”按钮,还可以把不需要的映射删除掉。

② 应用程序选项设置。主要设置应用程序的一些基本配置,如启用状态、缓冲、上层路径、默认的 ASP 程序等。设置操作方法并不复杂:用户只需在“应用程序配置”对话框中单击打开“应用程序选项”选项卡(图 4.22),从中即可设置。各设置项含义如下:

- “启用阶段状态”。选中该项即启用该应用程序,这样服务器与外部用户之间就可以相互交换数据。用户将数据请求发送给服务器,交由服务器中的应用程序进行处理,然后再返回给用户。从用户发出数据请求到收到处理后的数据,通常称为一个“阶段”。

- “阶段超时值”。用于设置用户与服务器之间交流数据的阶段的最大时间限度,以分钟为单位。对于不同的类型的交流阶段,所需时间有长有短。如果某个阶段时间太长,则必须断开这次通信阶段,并重新建立通信的连接。一般情况下,都可采用默认值。

- “启用缓冲”。选中该项即允许使用缓冲区。由于启用程序阶段的数据交换量比较大,所以通常要使用缓冲区来存放数据。这样一是可以加快数据的传输速率,二是可以缓解内

存资源的紧张程度。

- “启用上层路径”。即允许使用上一级目录，也就是父目录。
- “默认 ASP 语言”。用于设置默认的 ASP 应用程序语言，通常都是使用 VBScript 语言。

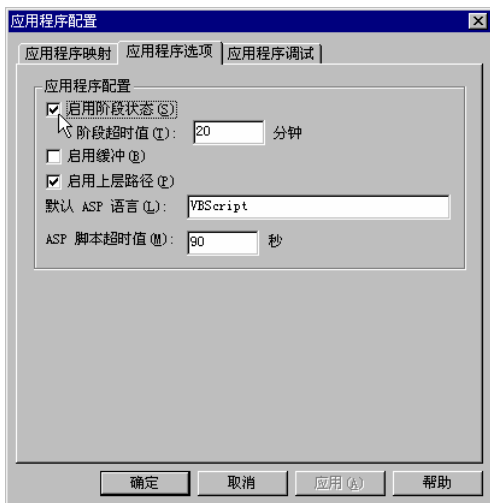


图 4.22 “应用程序选项”选项卡

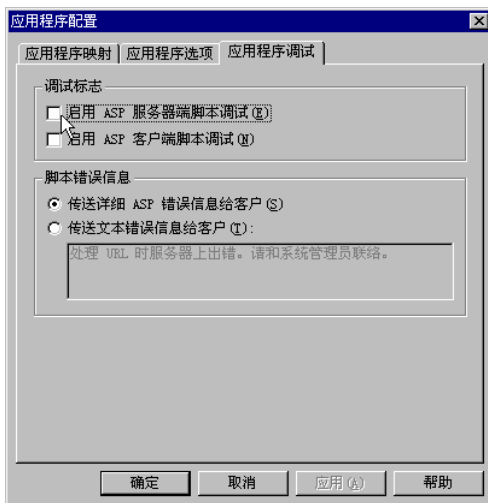


图 4.23 “应用程序调试”选项卡

- “ASP 脚本超时值”。即设置 ASP 程序在浏览器中的解释执行的最大时间限度。与一般的可执行程序不同，ASP 程序代码属于解释执行代码，不可以自动运行，必须经由其他程序解释，并翻译成可执行的命令，然后才能运行。



解释执行代码的程序有一个缺点，就是执行速度慢，程序效率低。

③ 应用程序调试功能设置。主要设置应用程序调试代码的各种选项。在“应用程序配置”对话框中单击打开“应用程序调试”选项卡，如图 4.23 所示，从中即可设置。

其中，“调试标志”区域用于设置有权调试 ASP 程序的某一方：用户方或服务器方。如选择“启用 ASP 服务器端脚本调试”一项，则允许服务器一方调试 ASP 程序代码；如选择“启用 ASP 客户端脚本调试”一项，则允许用户方调试 ASP 程序代码。

而“脚本错误信息”区域用于设置将代码错误信息发送给用户时的发送方式。若以 ASP 形式发送，则可选择“发送详细 ASP 错误信息给客户”一项，若以文本形式发送，则选择“发送文本错误信息给客户”一项。

综上，完成应用程序配置后，单击“确定”按钮，返回图 4.19 所示的“主目录”选项卡，再单击其中的“确定”按钮，即可完成主目录的设置。



注意，应用程序的配置是比较复杂的，不过大多数情况下我们直接采用默认设置即可，这样就无需配置应用程序了。

4.3.4 配置虚拟目录

主目录位置一旦改变，所有 Internet 用户的请求都将被路由到这个新的目录位置，IIS 也将把这个目录作为一个单独的站点来对待，并完成与各组件的关联。不过，有时 IIS 也可以把用户的请求指向主目录以外的目录，这种目录就称为虚拟目录。下面我们讲解虚拟目录是怎么回事。

我们知道，建站人员必须为建立的每个 Internet 站点都指定一个主目录。主目录是一个缺省位置，当 Internet 用户的请求没有指定特定文件时，IIS 将把用户的请求指向这个缺省位置。代表站点的主目录一旦建立，IIS 就会缺省地使这一目录结构全部都能由网络远程用户所访问，也就是说，该站点的根目录(即主目录)及其所有子目录都包含在站点结构(即主目录结构)中，并全部能由网络上的用户所访问。一般说来，Internet 站点的内容都应当维持在一个单独的目录结构内，以免引起访问请求混乱的问题。特殊情况下，网络管理人员可能因为某种需要而使用除实际站点目录(即主目录)以外的其他目录，或者使用其他计算机上的目录，来让 Internet 用户作为站点访问。这时，就可以使用虚拟目录，即将想使用的目录设为虚拟目录，而让用户访问。

处理虚拟目录时，IIS 把它作为主目录的一个子目录来对待；而对于 Internet 上的用户来说，访问时并感觉不到虚拟目录与站点中其他任何目录之间有什么区别，可以像访问其他目录一样来访问这一虚拟目录。设置虚拟目录时必须指定它的位置，虚拟目录可以存在于本地服务器上，也可以存在于远程服务器上。多数情况下虚拟目录都存在于远程服务器上，此时，用户访问这一虚拟目录时，IIS 服务器将充当一个代理的角色，它将通过与远程计算机联系并检索用户所请求的文件来实现信息服务支持。

创建虚拟目录的操作步骤如下：

(1) 在 Windows NT/2000 系统中启动 MMC(方法见上)，并从中展开 IIS 项，MMC 将显示 IIS 的配置选项。

(2) 使用鼠标右键单击想要配置其虚拟目录的站点，这会打开一个快捷菜单，从中单击“新增”子菜单下的“虚拟目录”命令，MMC 即会启动“新虚拟目录向导”并显示其对话




框。

(3) 在该对话框中的“用来访问虚拟目录的别名”字段框中为欲建的虚拟目录定义一个名字,如“newspaper”,然后单击“下一步”按钮继续。

(4) 在随后出现的对话框中,需要输入该虚拟目录的物理位置,即其实际位置。如果该虚拟目录在本地服务器中,则可直接指定路径,如“C:\mywebsite\myvirtualdirectory”;如果它在远程的服务器中,则需指定服务器名和虚拟目录的共享名(此时虚拟目录必须具有网络共享的属性),如“\\servename\sharename”。然后单击“下一步”按钮,新虚拟目录向导将显示设置访问权限的对话框。

(5) 从中为虚拟目录所代表的站点选择适当的权限。最后,单击“完成”按钮,所设虚拟目录即可生效。

以后,输入站点的 URL 地址并在后面依次加上斜杠(/)和虚拟目录的别名,即可访问该虚拟目录。例如,原站点的 URL 地址为 www.mywebsite.com,建立名为 newspaper 的虚拟目录后,输入 www.mywebsite.com/newspaper 即可访问这个虚拟目录。

使用虚拟目录的重要意义是,网络管理员可以把 Web 站点的负载分布到多台服务器上,这样使每台服务器都能保持较高的处理速度。

4.3.5 设置主页文件

在用户访问 Web 服务器的时候,如果要求他们必须记住并输入该站点的主页文件,这显然是不现实的。所以我们需要为服务器设置一个默认的主页文件,从而用户只需输入 Web 服务器站点的地址,便可以直接浏览该站点的主页,继而通过主页中的超链接去访问站点中的其他网页内容。在 IIS 的 Web 服务器中,一个站点的主页文件是可以任意设置的。具体操作方法如下:

(1) 在 MMC 中选取想要配置的 Web 站点并单击鼠标右键,然后从出现的快捷菜单中选择“属性”命令,即可打开当前所选站点的属性对话框(参见图 4.15 所示),从中单击并打开“文档”选项卡,如图 4.24 所示。

(2) 选中“启用默认文档”复选项,然后单击“添加”按钮,从出现的“添加默认文档”对话框(图 4.25)中输入主页文件名并单击“确定”按钮,即可把主页文件添加到“文档”选项卡中。



图 4.24 “文档”选项卡



图 4.25 添加默认文档

(3) 如果选项卡中的列表框内有多个主页文件，用户可单击左边的上、下箭头按钮，来调整它们的先后顺序，排在最上面的将是默认的主页文件，浏览器默认打开的就是该文件。

(4) 对于不必要的主页文件，用户还可以选中它后单击旁边的“删除”按钮，把它从列表中删除掉。

(5) 另外，如有需要，还可以在选项卡中指定是否需要“启用文档脚注”，以及指定脚注文件等。

(6) 最后单击“确定”按钮，即可使设置的主页文件生效。

4.3.6 目录安全设置

对站点而言，目录安全性是十分重要的。因为 IIS 中的目录其实就代表了站点，所以目录安全性也就是站点的安全性。目录安全性设置主要包括两部分工作，一是对访问服务器的计算机用户进行授权，二是限制某些外部用户的访问权。具体操作方法如下：

(1) 在 MMC 中选取想要配置目录安全性的 Web 站点并单击鼠标右键，然后从出现的快捷菜单中选择“属性”命令，打开当前站点的属性对话框(参见图 4.15 所示)，从中单击并打开“目录安全性”选项卡，如图 4.26 所示。

(2) 如需启用匿名访问及编辑它的验证方法，则可在“匿名访问和验证控件”区域单击“编辑”按钮，打开“验证方法”对话框，如图 4.27 所示。

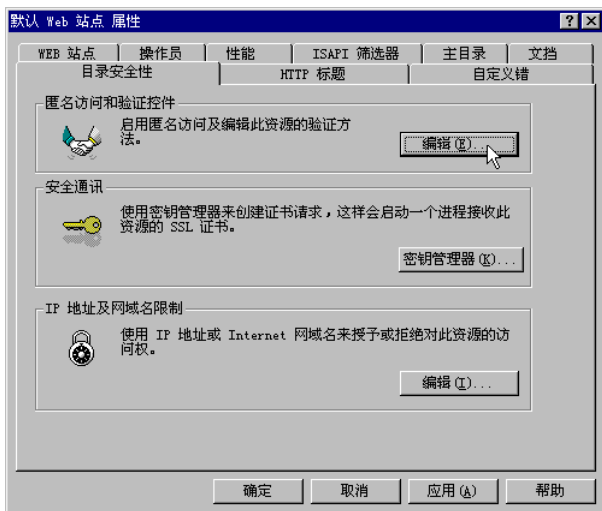


图 4.26 “目录安全性”选项卡

(3) 在该对话框中,选中“允许匿名访问”复选项,然后再单击“编辑”按钮,打开“匿名用户账号”对话框(图 4.28),从中即可输入或编辑匿名访问使用的账号,也就是“用户名”。

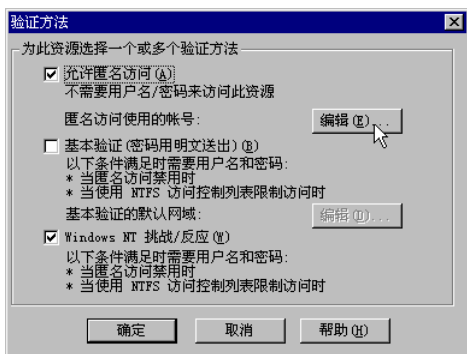


图 4.27 “验证方法”对话框

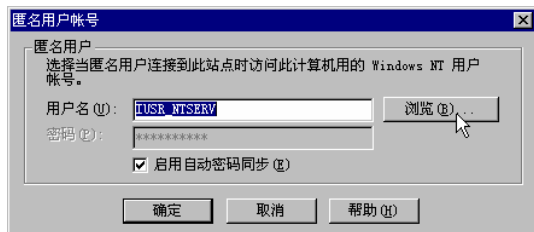


图 4.28 “匿名用户账号”对话框

(4) 如果一时想不起来用户名或用户名太复杂,则可单击旁边的“浏览”按钮,打开选择用户账号的对话框,如图 4.29 所示,从中选择需要的用户名,然后单击“确定”按钮,返回到“匿名用户账号”对话框。

(5) 一般来说,匿名用户账号是不设置密码的,所以需要选中对话框中的“启用自动密码同步”复选项;否则,就取消该项并在“密码”框内输入密码。设置完后单击“确定”按钮,返回“验证方法”对话框。

(6) 根据实际情况,在该对话框中还可以选择其他验证方法,如“基本验证”或“Windows NT 挑战/反应”等,具体设置比较简单,我们就不一一展开了。最后单击“确定”按钮,返回图 4.26 所示的“目录安全性”选项卡。

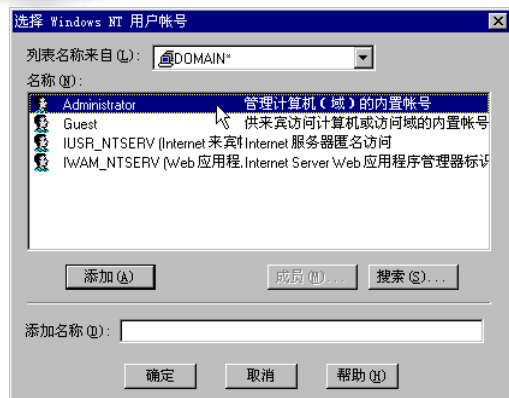


图 4.29 选择用户账号

(7) 若想实现安全通讯,则可使用密钥管理器来创建数字证书,方法是单击“安全通讯”区域里的“密钥管理器”按钮,打开服务器认证向导程序,然后按照向导程序指定的步骤执行,即可创建服务器的数字证书,完成服务器的完全验证。

(8) 此外,通过“IP 地址及网域名限制”区域内的“编辑”所打开的对话框,还可以限制一些用户或域访问服务器的权限。这样做的目的是防止外部用户一些破坏性的访问会对服务器造成危害,让指定的用户或域没有访问权,不能连接到服务器。

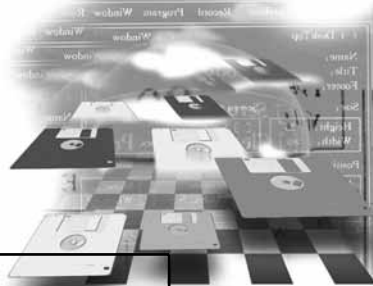
(9) 最后,单击选项卡中的“确定”按钮,即可完成目录安全性设置。

4.3.7 Web 站点负载的多台 IIS 服务器分布

有些站点的访问量是十分惊人的,比如 Microsoft 公司站点(www.microsoft.com)每天要接收 100 多万次的访问,这时就必须考虑 Web 站点的负载问题。这一问题的解决方案涉及较为复杂的技术问题,目前还没有一个十分理想的方案。常用的解决方案有两种,即虚拟目录法和循环 DNS 入口法。

正如我们前面讨论过的,IIS 服务器可以使用虚拟目录建立一个分布式目录结构,将出版的 Web 资源分布于多台其他服务器上。这样,在一定程度上可以把用户请求的负载分布到多台服务器之间,避免一台服务器必须处理所有用户请求的情况发生。不过,这种方法仅当站点的各组件被 Internet 用户均匀访问时才适用。如果用户针对站点的访问都指向某些特定的资源,比如有 80% 的访问都是技术支持主页,则使用虚拟目录的方法就难以奏效了。因为这时 IIS 服务器会简单地把大部分网络通信量集中到一台服务器上。那么,针对这种情况,我们需要尝试着使用第二种方法,即循环 DNS 入口法来解决。

一般而言,对某一特定主机,DNS 服务器只给它提供一个 DNS 入口,用于域名解析。由于来自远程 Internet 用户对站点的请求都要经过 DNS 服务器进行域名解析,所以专家们想到通过控制进入 DNS 服务器的请求来实现 Web 站点负载的分布。即为单台主机建立多个 DNS 入口,同时把每个入口分别指向不同的 IP 地址,即不同的服务器。比如,我们可以为



站点 `www.mywebsite.com` 指定多台服务器的 IP 地址：133.111.55.155、133.111.55.55、133.111.55.25 等。这样，当用户请求 DNS 服务器对 Web 站点的主机名进行 DNS 解析时，它将把第一个 IP 地址返回给用户；随后，DNS 服务器将为新的用户请求改变 DNS 入口的顺序，依次将第二个、第三个……IP 地址返回给用户；分配完可用 IP 地址后，再从第一个开始重新分配。如此反复，循环使用 DNS 入口分配用户的请求，从而解决 Web 站点的负载问题。这样做的前提条件是，网络管理员需要事先把 Web 站点的内容复制到 DNS 服务器所列出的与 IP 地址相应的多台 IIS 服务器上。

虽然循环 DNS 入口法能解决虚拟目录的不足，但它也不是最理想的方法，比如，有这么一种情况，那就是 DNS 服务器循环分配到某一 IIS 服务的用户请求恰好都是要执行同一任务，譬如下载大型文件，那么这台 IIS 服务器还是负载过重，影响各用户的使用效率。不过这种方法总比什么方法都没有要强，更好的方法还有待我们进行更深入的研究。

4.3.8 单站点服务器配置多个 Web 站点

如上所述，在多台 IIS 服务器之间分布 Web 站点负载，可以帮助降低任何一台 IIS 服务器的负载。但还有相反的情形，比如为了节省硬件资源，我们需要把多个 Web 站点放在同一台服务器上。

一般来说，一台 IIS 服务器通常只出版一个 Web 站点，这种服务器称为单站点服务器。事实上，在硬盘空间充足的情况下，单站点服务器可以同时容纳多个 Web 站点和 FTP 站点的内容，但 IIS 却无法单独管理多个站点。为此，有 3 种方法可以解决单站点服务器管理多个站点的问题：其一，在服务器上指定多个与站点相应的 IP 地址；其二，给每个站点使用不同的端口号；其三，使用主机首部名。

其中最好的方法是使用多个 IP 地址，因为它易于实现，且有着广泛的技术支持。指定不同端口号的方法需要网络管理员给各站点手工指定一个端口号，这是一种非标准方法，而且会增加用户访问站点的难度。主机首部名的方法仅适用于支持主机首部的 Web 浏览器，也就是说并不是所有的远程 Internet 用户都能满足这一方法的条件。

因此，建立多个 IP 地址实现单台服务器运行多个 Web 站点是首选方法。其配置操作包括两部分，一是先为运行 IIS 的 NT 服务器配置多个 IP 地址，二是为各站点配置相应的 IP 地址。第一部分操作的具体方法如下：

- (1) 在运行 Windows NT Server 系统的服务器桌面上，使用鼠标右键单击“网上邻居”

图标, 然后从打开的快捷菜单中选择“属性”命令, 打开“网络”对话框(图 4.30)。

(2) 选择并打开对话框中的“协议”选项卡, 如图 4.31 所示, 从协议列表中选择“TCP/IP”协议, 然后单击“属性”按钮, NT 会随之打开“TCP/IP 属性”对话框(图 4.32)。(3) 单击其中的“高级”按钮, 打开“高级 IP 寻址”对话框, 如图 4.33 所示。

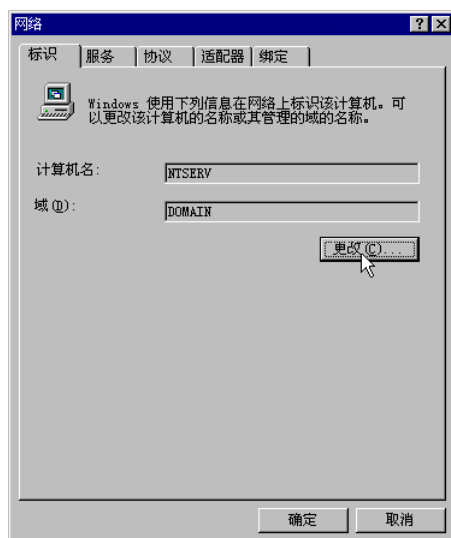


图 4.30 “网络”对话框

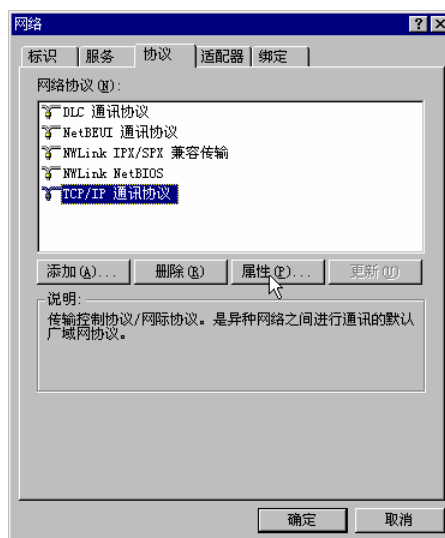


图 4.31 “协议”选项卡

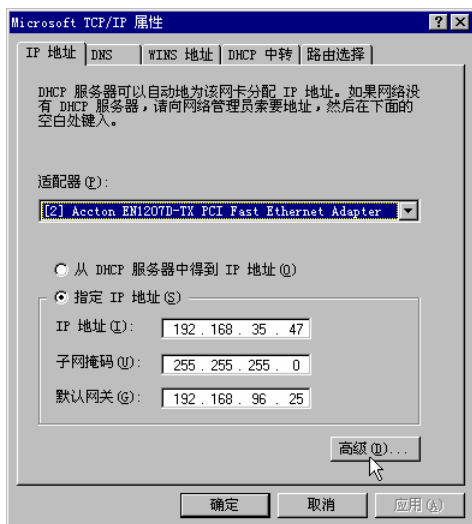


图 4.32 “TCP/IP 属性”对话框



图 4.33 “高级 IP 寻址”对话框

(4) 在该对话框中的“IP 寻址”框内,单击“添加”按钮,这时屏幕上会出现“IP 地址”对话框,如图 4.34 所示。

(5) 从中输入需要绑定到网络适配器上的新 IP 地址。然后,单击“添加”按钮,完成新 IP 地址的绑定工作,随后屏幕返回“高级 IP 寻址”对话框。如果需要绑定多个 IP 地址,则可重复上一步及本步操作,直到添加完所有的 IP 地址。



图 4.34 “IP 地址”对话框

(6) 单击对话框中的“确定”按钮,返回“TCP/IP 属性”对话框。

(7) 单击该框内“确定”按钮,返回“网络”对话框。这样就完成了把新 IP 地址绑定到网络适配器上的操作,此时系统提示网管重新启动计算机,以使新设置生效。

接下来,还要进行第二部分的操作,为各站点配置相应的 IP 地址。操作方法如下:

(1) 使用前面讲过的方法启动 MMC。在 MMC 中,选择想要给其分配 IP 地址的 Web 站点,并使用鼠标右键单击这个站点名,MMC 将会打开一个快捷菜单。

(2) 从中选择“属性”命令,屏幕上随之出现该站点的属性对话框。

(3) 选择该对话框中的“Web 站点”选项卡(参见图 4.15),然后单击打开“IP 地址”下拉列表,从中为当前站点选取一个 IP 地址。最后,单击“确定”按钮,MMC 即把对该 IP 地址的所有请求都指向上述选中的站点。

如果有多个 IP 地址需要配置,可重复上述操作,直到全部配置完。

4.4 建立和配置 WAP 站点服务器

完成 Web 站点的 WWW 服务器之后,我们就可以在该服务器的基础进一步配置,来建立 WAP 站点服务器了。通俗地讲,WAP 服务器首先是 WWW 服务器,它只不过比 WWW 服务器增加了对 WAP 的支持功能。那么,对于一台建立好的 WWW 服务器来说,需要增加哪些功能才可以提供 WAP 站点服务呢?由于 WAP 的客户端设备比较特殊,比如内存较小、显示屏幕也较小,所以 WAP 传输信息时不能采用普通的 HTML 文件格式,而是采用了基于 HDML 或 WML 的一种称为“卡片”的小型的文件格式(下一章会讲到),因此,我们只需为 WWW 服务器增加 WAP 的文件类型,实现对 WAP 文件的传输,即使 WWW 服务器变成一个 WAP 服务器,同时使 WWW 的 Web 站点变为 WAP 的站点。我们形象地把这一过程表示为:

WWW 站点服务器+WAP 文件类型=WAP 站点服务器

下面我们给出建立和配置 WAP 站点服务器的一般方法:

- (1) 按照上一节讲解的方法, 建立一个 Web 站点, 并将它配置为 WWW 服务器。该站点即将被设置为 WAP 站点, 所以不妨把它取名为“WAP”, 我们假设它的主目录名为“wap”。
- (2) 启动 MMC, 进入它的操作窗口(图 4.7)。
- (3) 从 MMC 左侧窗格中展开 Internet Information Server 组。MMC 将显示 IIS 服务器中的 Web 站点列表。
- (4) 选中“WAP”站点并单击鼠标右键, 然后从出现的快捷菜单中选择“属性”命令, 即可打开当前所选站点的属性对话框, 当前处于打开状态的是“Web 站点”选项卡(参见图 4.9 或图 4.15)。
- (5) 在该对话框中选择并打开“HTTP 标题”选项卡, 如图 4.35 所示。
- (6) 从其中的“MIME 映射”区域内单击“文件类型”按钮, 打开它的对话框, 如图 4.36 所示。可以看到, 该对话框中列出了一些已经注册的文件类型, 这些类型都是当前站点服务器所能支持的文件类型, 下面我们就为 WAP 传输增加文件类型。



图 4.35 “HTTP 标题”选择卡

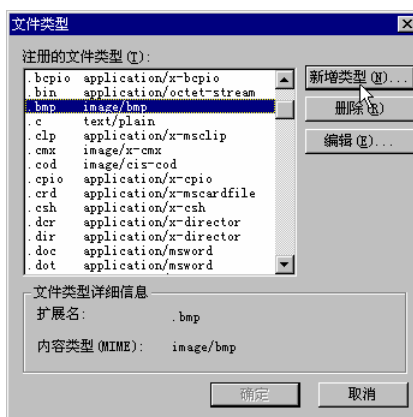


图 4.36 “文件类型”对话框

- (7) 单击对话框中的“新增类型”按钮, 屏幕上会出现一个小的编辑对话框, 从中即可输入 WAP 所需的文件类型。每个文件类型均有两个输入项目, 一个是扩展名, 另一个是内容类型。比如, 用于支持 WAP 的 WML 文件的扩展名为“.wml”, 内容类型为“text/vnd.wap.wml”, 将这两项输入到相应的文本框中, 然后单击对话框中的“确定”按钮, 即可把 wml 这一文件类型加入到当前服务器的注册文件类型列表中。

重复第(7)步操作,把 WAP 所需的其他文件类型也添加到当前服务器中。我们需要为 WAP 增添的文件类型主要有 6 种,表 4.2 给出了这些文件类型的扩展名和内容类型的描述。

表4.2 WAP所需的文件类型

扩展名	内容类型	说 明
.wbmp	image/vnd.wap.wbmp	WAP 支持的 1 位 bmp 位图
.wml	text/vnd.wap.wml	WAP 支持的 WML 语言的网页文本文件
.wmlc	application/vnd.wap.wmlc	WAP 支持的 WML 语言的应用程序文件
.wmlsc	application/vnd.wap.wmlscriptc	WAP 支持的 WMLScript 语言的应用程序文件
.wmlscript	text/vnd.wap.wmlscript	WAP 支持的 WMLScript 语言的网页文本文件
.wsc	application/vnd.wap.wmlscriptc	WAP 支持的 WMLScript 语言的应用程序文件

(8) 添加完后,如需要修改某一文件类型,可在图 4.36 所示对话框的注册文件类型列表中选中该类型,然后单击旁边的“编辑”按钮,从随后出现的对话框中即可更改该文件类型的扩展名和内容类型。如果要删除某一文件类型,可选中该类型后单击对话框(图 4.36)中的“删除”按钮即可。

(9) 最后单击对话框中的“确定”按钮,完成站点属性设置。这样我们就建立了一个真正的 WAP 站点,当前的 WWW 服务器即转变为真正的 WAP 服务器。如果 WAP 的客户端准备完好,我们就可以通过在 Winwap 等 WAP 浏览器上输入 `http://localhost(本地计算机名)/wap/index.html` 的形式,来访问和测试新建立的 WAP 站点了。当然,此前我们应当准备好 WAP 的主页文件 `index.html`,这恐怕需要等学习 WML 编程之后才能编写,更详细的测试工具和编程方法我们会在以后各章中给出。

4.5 PWS 的安装设置与 WAP 服务器配置

前面提到,对许多学习 WAP 的普通用户来说,可能不具备局域网环境,不能使用 IIS 来建立 Web/WAP 服务器。那么怎么办呢?通常的解决办法是使用 Microsoft 公司专门为这种情况提供的个人 Web 服务器系统 PWS(Personal Web Server)来建立 Web 服务器,并进而建立 WAP 服务器。PWS 的安装与 IIS 的安装类似,配置操作更为简单,我们不再叙述详细的操作步骤,只是给出比较关键的操作。

4.5.1 PWS 的安装

PWS 一般随正版的 Windows 98 系统一同提供。安装时可进行如下操作：

(1) 在 CD-ROM 驱动器中放入 Windows 98 光盘，运行 X:\add-ons\PWS 目录下的 setup.exe 程序(其中 X 为光驱对应的盘符)。然后在出现的安装画面中选择安装方式：“最小安装”、“典型安装”或“自定义安装”。前两种都是比较简单的安装，我们以“自定义安装”为例说明。

(2) 在选择“自定义安装”并单击“下一步”按钮后，将出现图 4.37 所示的组件列表对话框让用户选择所要安装的组件。

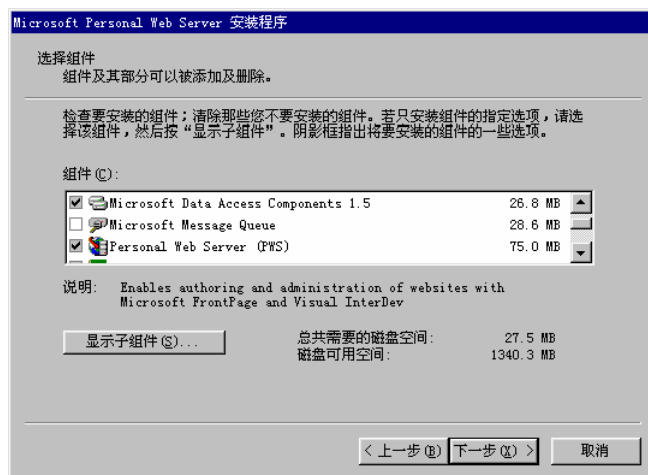


图 4.37 选择组件

(3) 单击选中“组件”列表中的“Microsoft Data Access Components 1.5”一项，然后单击“显示子组件”按钮，从出现的对话框中选取所有的子组件，以及每个子组件的所有子组件等。

(4) 采用同样的方法改变“组件”列表中“Personal Web Server(PWS)”一项的默认值，选中其中所有子组件。最后单击“下一步”按钮。安装向导将给出图 4.38 所示的对话框让用户设置 Web 发布的主目录。

(5) 一般情况下，可直接采用默认的主目录。如有必要，也可以在“WWW 服务”文本框中键入需要的 Web 发布主目录的物理路径。然后，单击“下一步”按钮，系统将出现让用户设置 MTS 安装文件夹的画面。一般来说，接受系统的默认配置即可。再单击“下一步”

按钮，随后系统将开始安装 PWS。



图 4.38 设置主目录

(6) 安装完 PWS 后，重新启动计算机，就可以发现“程序”菜单的“Internet Explorer”子菜单中增加了一条“Personal Web Server”命令，它是一子菜单，含有 PWS 的几条命令，如图 4.39 所示。接下来，用户就可以设置和使用 PWS 了。

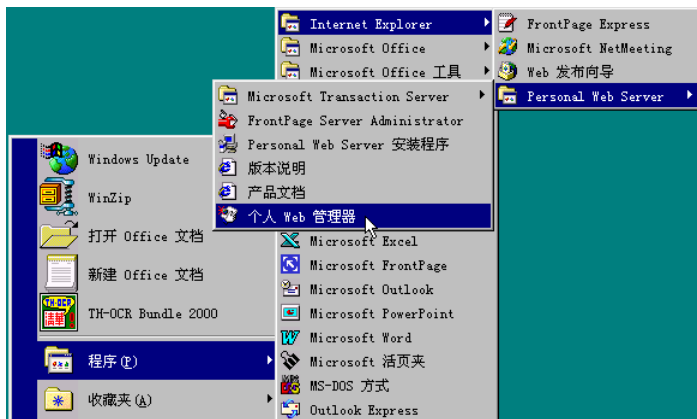


图 4.39 PWS 的菜单命令

4.5.2 PWS 的设置

对大多数用户来说，PWS 的设置主要包括两项内容，一是设置 IP 地址，二是设置目录属性。设置 IP 地址的操作方法如下：

(1) 在 Windows 98 系统的桌面上，使用鼠标右键单击“网上邻居”图标，然后从打开的快捷菜单中选择“属性”命令，打开“网络”对话框，如图 4.40 所示。

💡 如果桌面上没有“网上邻居”图标，则可单击“开始”按钮，打开“开始”菜单，从其“设置”子菜单中选择“控制面板”命令，双击其中的“网络”项，Windows 也会打开图 4.39 所示的“网络”对话框。

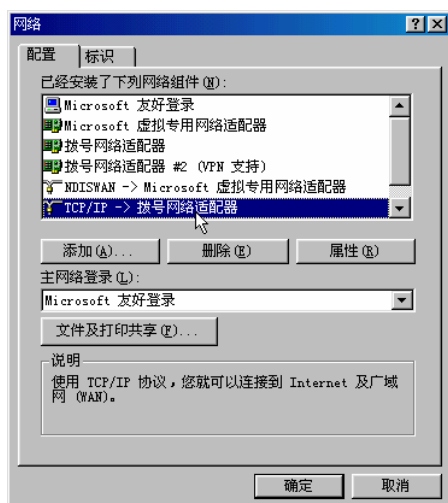


图 4.40 “网络”对话框

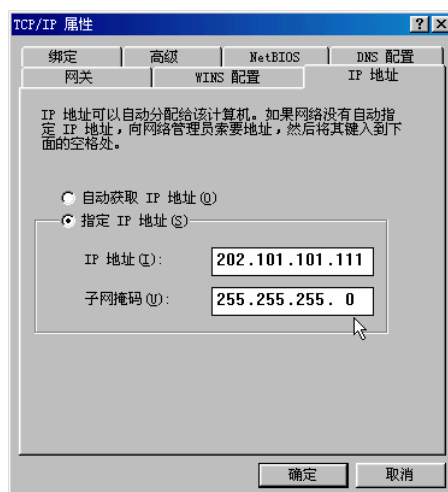


图 4.41 “TCP/IP 属性”对话框

(2) 从“配置”选项卡的协议列表中选择“TCP/IP”协议，然后单击“属性”按钮，打开“TCP/IP 属性”对话框，如图 4.41 所示。

(3) 从中选择打开“IP 地址”选项卡，然后在“IP 地址”栏中输入任意一个 IP 地址，如 202.101.101.111，在“子网掩码”中输入 255.255.255.0，最后单击“确定”按钮，即可完成 IP 地址的设置。该地址仅供内部测试使用，所以不一定使用授权的 IP 地址。

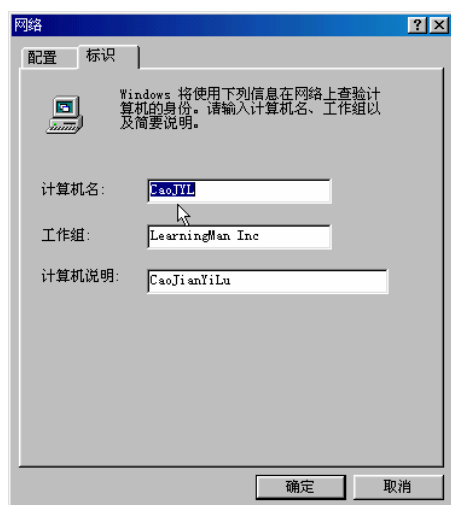


图 4.42 设置计算机名

(4) 再从“网络”对话框中单击打开“标识”选项卡，如图 4.42 所示。用户可根据个人喜好，在“计算机名”文本框中输入计算机名称，然后单击“确定”按钮，关闭“网络”对话框并重新启动计算机，所做设置即可生效。

所设计算机名用于内部识别，当进行浏览器与服务器的测试时，还可用于 URL 定位。如需检查 IP 地址的设置是否正确，可以在浏览器的地址栏中键入“http://计算机名或 IP 地址”，如果设置正确的话，

就会显示缺省主页。

完成 IP 地址设置后,用户还需要设置目录属性。在 PWS 中只能设置主目录的属性,其他目录的属性则与主目录的属性一致。设置主目录属性的步骤如下:

(1) 单击“开始”按钮,将光标依次指向“程序”→“Internet Explorer”→“Personal Web Server”→“个人 Web 管理器”。启动个人 Web 管理器,打开它的工作窗口。

(2) 单击其中的“高级”图标,在右边窗格中将出现高级选项窗口,如图 4.43 所示。接下来我们就可以在这个窗口中设置主目录的属性了。

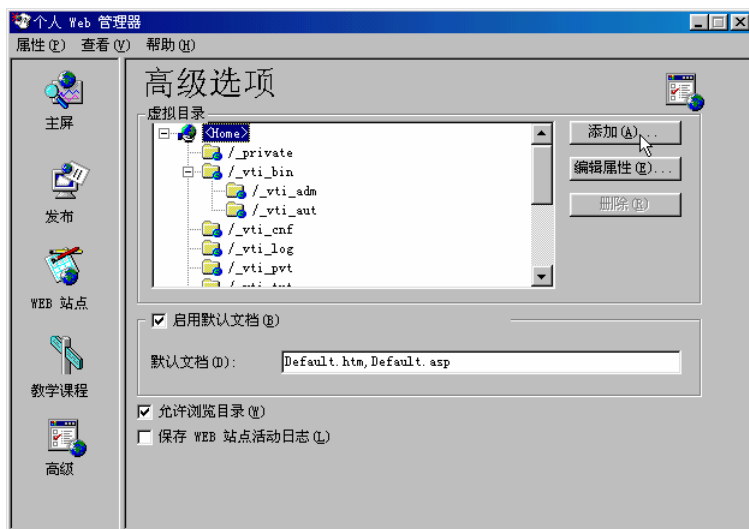


图 4.43 个人 Web 管理器的“高级”选项

(3) 设置虚拟目录。单击选中“<Home>”,然后单击“编辑属性”按钮,打开“编辑目录”对话框,如图 4.44 所示。

(4) 在“目录”框中输入主目录所对应的物理路径,系统默认的缺省路径为“C:\Inetpub\wwwroot”。也可单击“浏览”按钮,从打开的对话框(图 4.45)中选择需要的目录。

(5) 对话框中的“访问”一栏中有 3 个项目,主要用于设置用户对 ASP 文件的访问权限。其中,“读取”用于设定用户有权查看网页文件的内容,为了安全起见,我们建议不要赋予用户这一权限;“执行”用于设定用户有权执行 ASP 编写的网页文件;“脚本”用于设定可以执行脚本语句。要赋予用户相应的权限,只要单击选中该选项即可。最后单击“确定”按钮,即可返回个人 Web 管理器的“高级”选项窗口。

(6) 启用默认文档。在“高级”窗口选取该项,表示如果用户请求网页时只输入了网址

或网址和目录，而没有指定具体的网页文件，浏览器将加载默认文档即主页。用户可在“默认文档”后的文本框中输入该默认文档的名称。如果文档有多个，则各文档名之间用逗号隔开。服务器在接到没有指定文件名的网页请求时，将按次序搜寻本目录下的默认文档，然后将第一个找到的默认文档传送到浏览器。

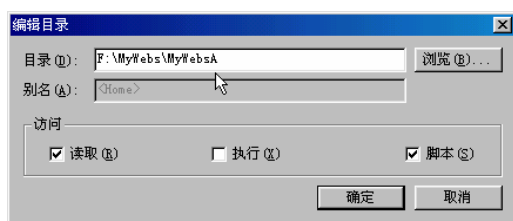


图 4.44 “编辑目录”对话框



图 4.45 选择文件夹

(7) 允许浏览目录。在“高级”窗口中选取该项表示当没有启用默认文档，或服务器找不到规定的默认文档时，将返回该目录下所有子目录的文件的超文本列表。为了网站的安全，建议用户不要选取此选项。

(8) 保留 Web 站点活动日志。在“高级”窗口中选取该项可建立日志文件来保存用户访问网站的活动情况。

完成这些设置后，用户就可以正式使用 PWS 来发布站点和用浏览器访问该站点了。不过这里建立好的站点属于 WWW 的站点，PWS 承担的服务器角色也是 WWW 服务器，而不是我们需要的 WAP 服务器，下面我们讲解将 PWS 的 WWW 服务器变成 WAP 服务器的具体方法。

4.5.3 设置建立 WAP 服务器

与 IIS 系统中类似，把 PWS 系统的 WWW 服务器转变成 WAP 服务器也需要增加 WAP 的文件类型。由于 PWS 系统没有提供增加文件类型的途径，所以我们需要通过设置 Windows 98/2000 的注册表或文件选项来增加 WAP 的文件类型。下面我们就分别介绍这两种方法。

通过设置注册表来建立 WAP 服务器

具体操作方法如下：

(1) 选择“开始”菜单中的“运行”命令，从出现的对话框的命令行中输入“Regedit.exe”，然后单击“确定”按钮，启动“注册表编辑器”程序，如图 4.46 所示。

(2) 下面我们增加 WAP 所需的 wml 文件类型的扩展名“.wml”及其内容类型“text/vnd.wap.wml”。首先从注册表中选中“HKEY_CLASSES_ROOT”，然后单击鼠标右键，从出现的快捷菜单中选择“新建”子菜单中的“主键”命令，如图 4.46 所示。

随后出现图 4.47 所示的“新键#1”的提示，在此处输入主键“.wml”并予以确认。接下来，再选中新加入的主键，并单击鼠标右键，打开它的快捷菜单(同图 4.46)，从中选择“新建”子菜单中的“串值”命令，输入串值名称为“Content Type”，然后双击该名称，打开它的编辑对话框，如图 4.48 所示，从中输入它的键值“text/vnd.wap.wml”，随后单击“确定”按钮，完成主键的添加操作。

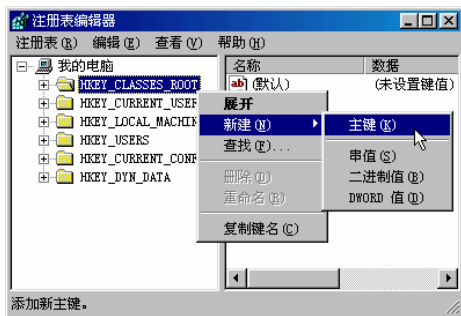


图 4.46 注册表操作



图 4.47 在 HKEY_CLASSES_ROOT 层加入主键

(3) 再从注册表中展开到“HKEY_LOCAL_MACHINE\Software\CLASS ES\MIME\ Database\Content Type\”，如图 4.49 所示。采用上述方法，为“Content Type”添加主键“text/vnd.wap.wml”，并加入它的串值名称



图 4.48 输入键值

“Extension”和键值“.wml”。至此，为 WAP 增加 wml 文件类型的操作才算完成。

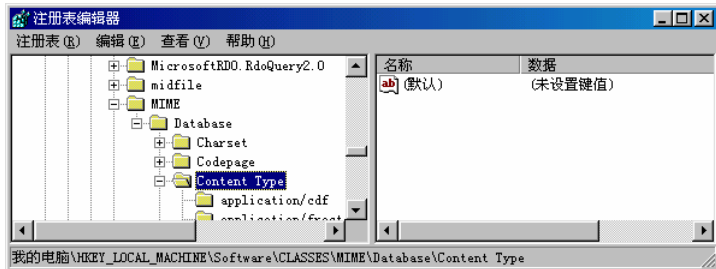


图 4.49 添加 Content Type 的主键和串值

(4) 重复上述两步操作，为 WAP 添加所需的其它几种文件类型。各种文件类型的扩展名及内容类型可参见表 4.2。

完成这些操作后，当前 PWS 系统的服务器站点即可充当为 WAP 站点，并可承担 WAP 服务器的工作了。

通过设置文件选项来建立 WAP 服务器

设置文件选项的目的也是为 PWS 系统增加 WAP 所需的文件类型。操作方法如下：

(1) 打开“Windows 资源管理器”，从其“查看”菜单中选择“文件夹选项”命令，打开它的对话框，并选择“文件类型”选项卡，如图 4.50 所示。



图 4.50 “文件类型”选项卡

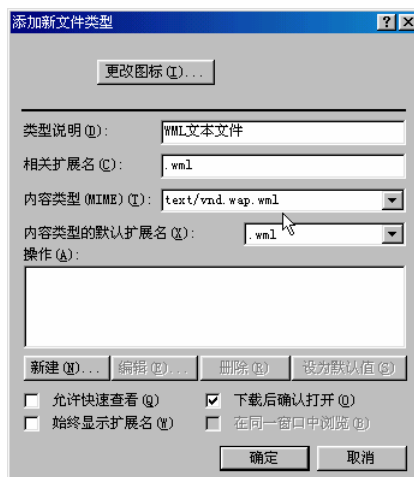


图 4.51 设置 WAP 所需的文件类型

(2) 单击其中的“新类型”按钮，屏幕上就会出现“添加新文件类型”对话框，如图 4.51 所示。从中即可输入 WAP 所需文件类型的扩展名和内容类型，而且还可以加上类型说明。图 4.51 中我们就输入了 wml 类型的扩展名.wml 和内容类型 text/vnd.wap.wml。输入完后单击“确定”按钮，即可把该类型添加到图 4.50 所示的文件类型列表中。

(3) 重复上述操作，把 WAP 所需的表 4.2 所示的几种文件类型全都添加到文件类型列表中。这样就把当前 Windows 系统中 PWS 的 WWW 服务器转变成了 WAP 服务器。



4.6 建立免费个人 WAP 网站的简易方法


正如我们可以在普通 Internet 环境中建立个人免费站点一样，我们也可以建立免费的个人 WAP 网站，发布基于 WAP 的个人信息甚或服务，而且方法类似。目前，Internet 上有许多站点提供有个人免费 WAP 站点服务，用户可以访问它们的网站，按照它们给出的提示，完成必要的设置项目，即可建立自己的 WAP 网站。

例如，我们可以到 <http://www.hoiley.com/> 这个网站建立个人 WAP 网站。进入该网站后要先申请一个用户名，比如 caowap，然后使用该用户名登录，即可进入建立 WAP 站点的控制页。该页面中有一排功能导航条，单击其中的“Create a WAPpage!”按钮，屏幕上就会出现一个对话框，要求我们输入 WAP 页面的名称，比如输入“mywaphome”，按回车后 mywaphome.wml 的链接就会出现在控制页的上方。

此时，单击 mywaphome.wml 超链接即可进入该页的编辑界面，从中填上页面标题以及需要显示的内容即可生成一个简单的个人 WAP 主页。注意其中有一个“Upload”的图标按钮，单击它我们可以上载 wbmp 格式的图片，这样就可以制作一个图文并茂的 WAP 主页。

如果需要，还可以制作更下一级的 WAP 网页并连接到主页上，最后即可建立好个人的 WAP 网站。

以后，使用 WAP 的模拟浏览器或具有 WAP 功能的手机(如 Nokia7110 手机)，通过输入网址“[http://www.hoiley.com/user/caowap\(mywaphome.wml\)](http://www.hoiley.com/user/caowap(mywaphome.wml))”就可以访问这个免费的个人 WAP 站点了。

 需要指出的是，WAP 网页只支持 1 位的 bmp 位图，即单色的 bmp 图像，普通图像不能直接应用到 WAP 页面中，需要使用一些专门的工具软件进行格式转换，将图像转换为 WAP 支持的 wbmp 格式，然后才能应用到 WAP 页面中。我们将在第 11 章介绍图像格式的转换工具及转换方法。

本章小结

本章在介绍 Web 服务器一般构建方法的基础上，讲解了安装和使用 IIS、PWS 建立 WWW 服务器，并进而建立 WAP 服务器的具体方法。本章内容技术性比较强，读者最好实



际安装和测试一下 IIS 或 PWS，以便能够全面掌握 Web/WAP 服务器的建立与配置方法。

虽然说本章介绍的是 WAP 编程之外的两个软件，但如果没有这两个软件所建 WAP 服务器的支持，我们就不可能实现 WAP 的站点建设功能，也就不能测试后面学习的 WAP 编程技术。因此，希望大家不要忽视本章内容的学习，尤其要掌握把一个 WWW 服务器变成 WAP 服务器时需要增添的文件类型(见表 4.2)。



第 5 章 WML 语言基础

完成 WAP 服务器的建立和 WAP 浏览器的安装之后，我们接下来就可以使用 WML 语言来编写 WAP 网页或应用，并通过 WAP 服务器及浏览器进行调试了。从本章开始我们将系统地学习 WML 语言，本章主要讲解 WML 语言的基础知识，下一章全面讲解 WML 的语法、标签和规则。

5.1 WML 的简单例子及编辑、测试方法

无线标记语言 WML(Wireless Markup Language)是一种基于扩展标记语言 XML(Extension Markup Language)的语言，是 XML 的子集。它可以显示各种文字、图像等数据，是由 WAP 论坛(<http://www.wapforum.org/>)提出并专为无线设备用户提供交互界面而设计的，目前版本是 1.1 版。这些无线设备包括移动电话、呼机和个人数字助理 PDA(Personal Digital Assistants)等。

5.1.1 WML 与 WAP 设备

为了更好地理解和使用 WML 语言，开发人员应对 WML 适用的设备和支持 WML 的设备的特点、特征有个大概的了解。

一般而言，WML 适用的无线设备通常具有以下特点：

- 与普通的个人计算机相比，体积较小；
- 设备的内存有限，且其 CPU 性能也有限；
- 通讯带宽较窄、时延较长。

以移动电话、PDA 为例来讲，支持 WML 的设备主要具有以下特征：

- 有一个显示屏幕，可以显示 4 行字符，每行 12 个字符；4 行字符中通常包括保留给

功能按钮的那一行;

- 支持数字和字符的输入;
- 支持垂直和水平滚动的箭头按键;
- 支持操作者使用箭头或数字按钮进行选择;
- 支持 ASCII 的可打印码;
- 通常都有两个可编程功能键, 即 Accept 键和 Options 键, 一般安排在接近键盘的屏幕下方;
- 通常有一个 Prev 导航键。

我们介绍 WML 所适用 WAP 设备的目的, 是希望读者通过 WAP 设备的特点、特征来了解 WML 语言的特点, 进而对 WML 编程所要解决的问题有个大概的认识。

5.1.2 使用文本编辑器编写 WML 程序

使用 WML 语言编写 WAP 网页或应用时, 需要使用一个编辑器进行编辑。与 HTML 编程一样, WML 编写的程序也是纯文本文件, 可以使用任一文本编辑器进行编写, 比如 Windows 系统中的“记事本(NotePad)”等, 也可以使用前面我们介绍的 WAP 开发工具包的编辑器进行编写, 比如 Nokia WAP Toolkit 等。我们先介绍第一种方法, 随后介绍第二种。

如果要使用“记事本(NotePad)”来编写 WML 程序, 则可在 Windows 95/98 系统中, 单击“开始”按钮, 然后从出现的菜单中, 依次将光标指向“程序”、“附件”、“记事本”, 启动“记事本”程序。屏幕上随后就会出现它的编辑窗口, 从中就可以输入并编写 WML 程序了。

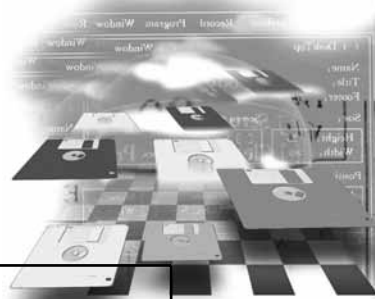
作为举例, 我们可以输入如下简单的程序:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/
wml_1.1.xml">

<wml>

    <card id="card1" title="Title">

        <p>
```



```
<!-- Write your card implementation here. -->
Hello World!!!

</p>
</card>

</wml>
```

输完后将它保存为 `hello.wml` 文件。保存时注意文件的扩展名应为 `.wml` 而不是 `.txt`。


5.1.3 使用微浏览器测试 WML 程序

输入完上述 `hello.wml` 程序后，大家一定想查看一下它的运行效果。那么使用什么工具呢？最直接的工具是 WAP 手机，但这需要 WAP 服务供应商的支持，对大多数开发人员来说不一定能实现。最方便的工具则是微浏览器，比如 WinWap 等。我们这里就以 WinWap 浏览器为例，说明测试 WML 程序、观察 WML 网页运行效果的操作方法。

通过上一章的学习，我们已经掌握了 WAP 服务器的建立方法，而且我们还知道，WAP 服务器是在 IIS 系统下的 WWW 服务器中建立的。通过在 WWW 服务器中建立一个名为“wap”的虚拟目录，我们就实现了 WAP 服务器。

为了测试 `hello.wml` 程序的效果，首先我们需要将该程序复制到 WAP 服务器中，也即 WWW 服务器的 wap 目录中。这一过程，我们通常称为 WAP 网页的发布。

然后，在确保 WWW 服务已经启动的情况下，启动 WinWap 微浏览器，并在 URL 栏里输入 `http://127.0.0.1/wap/hello.wml`，并按下回车。即可在 WinWap 的浏览窗口中查看 `hello.wml` 页面的运行效果。如图 5.1 所示，`hello.wml` 程序在窗口中显示一个名为“Title”的标题，及一句“Hello World!!!”的文字。这个效果其实也是使用 WAP 手机访问该网页时得到的效果。

需要说明的是，“`http://127.0.0.1/wap/hello.wml`”中的“127.0.0.1”是指 WWW 服务器所在计算机的默认 IP 地址，我们也可以使用该计算机的识别名来代替它。比如，若该计算机名为“caojianwap”，则输入“`http://caojianwap/wap/hello.wml`”也可访问上述 WAP 网页。

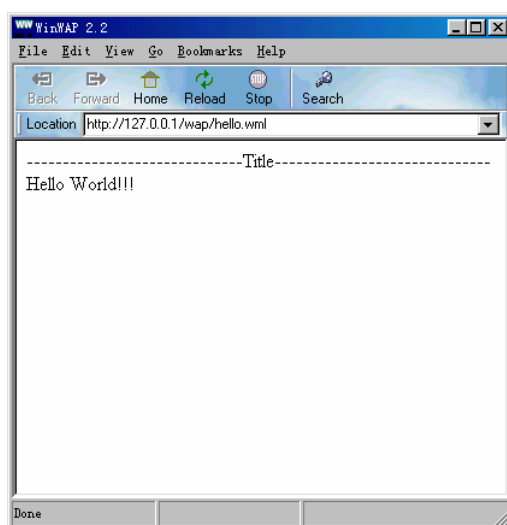



图 5.1 WinWAP 微浏览器浏览 hello.wml 页面

5.1.4 使用 WAP 开发工具包编辑和测试 WML 程序

我们以 Nokia WAP Toolkit 为例，说明使用 WAP 开发工具包编辑和测试 WML 程序的操作方法。首先，我们启动 Nokia WAP Toolkit，然后在其编辑窗口中输入上述 hello.wml 程序，如图 5.2 所示。如果输入过程中出现错误或需要进一步修改程序，可在该编辑窗口中直接操作。

然后，单击该编辑窗口下方的“编译(Compile)”按钮，即可保存并编译当前窗口中的 WML 程序。随后再单击旁边的“显示(Show)”按钮，即可运行窗口中的程序，并把运行结果显示在 Nokia 的 WAP 手机模拟器窗口中，如图 5.2 右边的 Nokia 手机屏幕中的显示信息所示。这一结果与使用实际的 Nokia 手机访问 hello.wml 网页时得到的结果完全一致。

显然，使用 WAP 开发工具包编辑和测试 WML 程序，要比使用文本编辑器编辑，再用 WinWap 测试 WML 程序的操作简便得多，快捷得多。以后编辑和测试 WML 程序时，建议大家采用某一种 WAP 开发工具包工作，这种集成化的操作环境，必定可以大大提高我们的开发效率。

 需要强调指出的是，WAP 开发工具包通常还提供了许多辅助开发工具或编辑、调试功能。以 Nokia WAP Toolkit 为例来说，它在提供程序编辑窗口的同时，还提供了“消息(Messages)”、“变量(Variables)”、“历史(History)”、“书签(Bookmarks)”等选项卡(参见图

5.2), 通过这些选项卡, 可以查看 WML 程序编译或调试过程中出现的有关信息, 或查看、编辑程序中的变量、书签, 以及查看历史操作等。而且还提供了菜单命令, 通过这些菜单命令可以对开发工具系统进行有关配置。所有这些操作都不复杂, 所以我们本书就不详细展开了。不过实际开发中, 大家应试着学习和使用这些选项卡、功能及命令, 以优化开发环境和提高开发效率。

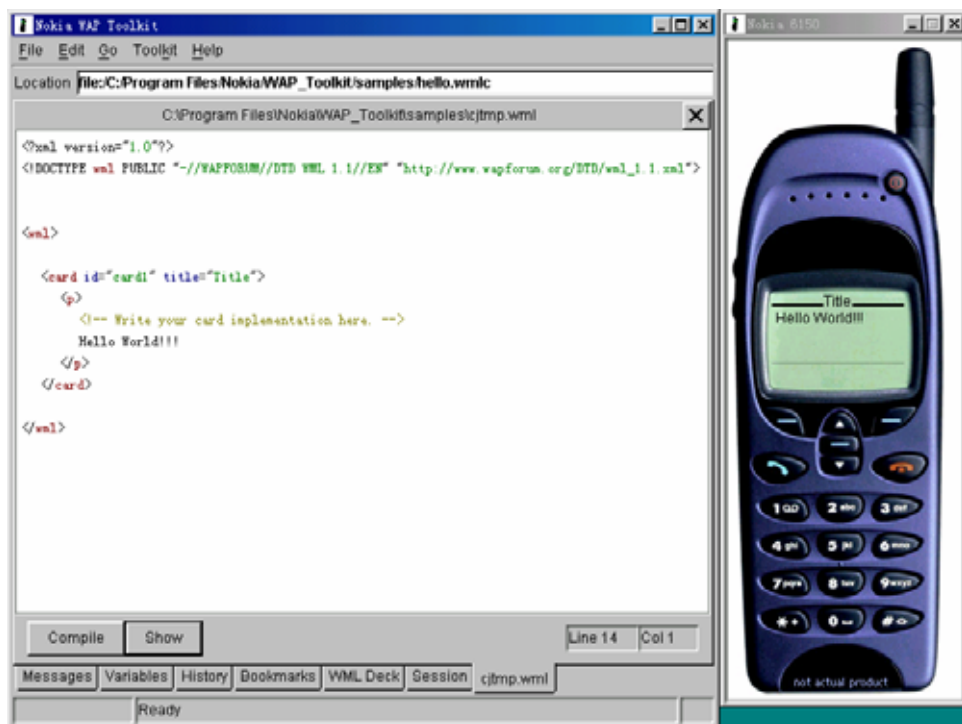


图 5.2 使用 WAP 开发工具包编辑和测试 WML 程序

5.2 WML 程序结构

上一节我们已经接触了一个简单的 WML 程序, 具有 HTML 编程基础的读者可以看出, WML 程序在结构形式上与 HTML 程序有很多相似之处。下面我们就根据一个实例来分析一下 WML 程序的结构及组成。

5.2.1 WML 的元素和标签

分析实例之前, 我们有必要对 WML 的元素和标签予以简单说明。与 HTML 类似, WML

的主要语法也是元素和标签。元素是符合 DTD(文档类型定义)的文档组成部分,如 title(文档标题)、IMG(图像)、table(表格)等等,元素名不区分大小写。WML 使用标签来规定元素的属性和它在文档中的位置。标签使用小于号(<)和大于号(>)括起来,即采用“<标签名>”的形式。标签分单独出现的标签和成对出现的标签两种。大多数标签是成对出现的,由首标签和尾标签组成。首标签和尾标签又分别称为起始标签和终止标签。首标签的格式为“<元素名>”,尾标签的格式为“</元素名>”。成对标签用于规定元素所涵的范围,比如和标签用于界定黑体字的范围,也就是说,和之间包住的部分采用黑体字显示。单独标签的格式为“<元素名/>”,它的作用是在相应的位置插入元素。如
标签表示在该标签所在位置插入一个换行符。

5.2.2 WML 程序结构形式及组成的实例分析

了解了上述知识后,下面我们再分析一个实例程序。程序如下:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1/EN" "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>

  <card id="card1" ontimer="#card2" title="Toolkit Demo">
    <timer value="50"/>
    <p align="center">
      <br/> <br/> <br/>
      <big>
        <!-- Write your card implementation here. -->
        Welcome to ...
      </big>
    </p>
  </card>

  <card id="card2" ontimer="#card3" title="Toolkit Demo">
    <timer value="50"/>
    <p align="center">
```




```

        <br/> <br/>
        <b>
            The Nokia<br/>
        </b>
        <u>
            Wireless Application Protocol
        </u>
        ...
    </p>
</card>

<card id="card3" title="Toolkit Demo">
    <p align="center">
        <br/> <br/> <br/>
        <big>
            <i>
                Toolkit!
            </i>
        </big>
    </p>
</card>

</wml>

```

该程序运行后将在 WAP 手机屏幕上依次显示 3 屏信息。先显示 “Welcome to ...”，然后显示 “The Nokia Wireless Application Protocol...”，最后显示 “Toolkit!”。显示时每屏都有标题 “Toolkit Demo”，相邻两屏之间延时为 50，其单位大小为 1/10 秒，延时 50 即 5 秒。

通过以上示例我们可以了解到 WML 程序的结构形式及组成：

(1) 语法。WML 的语法与 HTML 极为相似，仍然是一种标记语言，并且延续了 XML 的语法规则。具体的语法规则我们后面会详细介绍的。

(2) 文件声明。所有的 WML 程序必须在文件的开头处声明 XML 文件类型，包括 XML 的版本，WML 的文档类型、所用规范等。声明形式如下：

```

<?xml version="1.0"?>

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

```

(3) 标签。在 WML 语言中需要使用标签(Tag), 其使用形式与 HTML 和 XML 等标记语言中的形式是完全一致的。

(4) 元素。WML 的元素(Element)用于描述卡片组(Deck)的标记信息及结构信息。一个元素通常由一个首标签、内容、其他元素及一个尾标签组成, 具有下述两种结构之一:

<首标签> 内容 </尾标签>

或

<标签/>

元素包含的内容中还可以有元素, 这些元素也是由首标签、相应内容、其他元素及尾标签组成。不包含内容的元素称为空元素, 它退化成一个单独的标签。或者说, 单独的标签也是一个元素。

(5) 属性。WML 与 XML 一样, 其标签可以包含很多属性。属性用于给标签提供必要的附加信息, 且属性内容通常在起始标签内使用。不过, 属性内容不会被浏览器显示, 它只作为参数为标签提供必要的信息。

指明属性值的时候, 需要把该值用引号括起来, 可以是单引号或者双引号, 引号通常成对嵌套使用。属性名称必须小写。例如: <card id="card1" ontimer="#card2" title="Toolkit Demo">。

而且, 单引号的属性中还可以包含双引号的属性。实体字符也可作为属性值。实体字符是指诸如&、<、>、'、"的特殊字符, 在 WML 程序中显示这类字符需要特殊处理, 后面我们介绍具体方法。

(6) 注释。WML 程序中也可以加入注释。注释内容用于给开发人员顺利阅读源代码提供方便, 它不会被浏览器显示出来。注释内容在标签中以感叹号(!)引出, 并用小于号(<)和大于号(>)括起来, 注释内容两端各加两个减号, 即采用<!--注释内容-->的形式。例如: <!--Write your card implementation here. -->。需要说明的是, WML 程序中不支持注释的嵌套。

(7) 文档结构。WML 文档是由“卡片(Card)”和“卡片组(Deck)”构成的, 一个 Deck 是一个或多个 Card 的集合。当客户终端发出请求之后, WML 即从网络上把 Deck 发送到客户的浏览器, 这时用户就可以浏览 Deck 内包含的所有 Card, 而不必从网上单独下载每一个 Card。程序中的第一个 Card 是缺省的可见的 Card。



注意, Deck 是一副纸牌的意思, 这里是指一叠卡片, 所以我们本书把它翻译为“卡



片组”。另外，Card 指的是 WAP 手机屏幕大小的网页，尽管有时一个 Card 可能需要多屏才能显示完，但我们也可以把它翻译为“页面”，不过这样与 HTML 中的“页面”不易区分，所以我们还是采用“卡片”的译法。

5.2.3 WML 程序的基本结构

以上我们简单分析了 WML 的程序结构及组成，由此大家可以对 WML 程序有个整体上的初步认识。下面我们给出 WML 程序的基本结构：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.
org/DTD/wml_1.1.xml">
<wml>
  <head>
    <access/>
    <meta.../>
  </head>
  <card>
    Some contents...
  </card>
</wml>
```

该基本结构可以分为以下几个关键部分：

(1) 声明。WML 程序由许多 Deck 组成，对于每一个 Deck，在其文档开头必须进行 XML 的声明和文档类型 DOCTYPE 的声明。

XML 声明总是在文件的第一行，注意前面最好不要有空格或者换行：

```
<?xml version="1.0"?>
```

紧跟着是 DOCTYPE 声明，注意声明时字母的大小写不要搞乱：

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

(3) <wml>标签。该标签用于包含和定义 WML 的一个 Deck。它有一个可选的 xml:lang 属性来制定文档的语言，比如<wml xml:lang="zh">表示文档语言为中文。

(4) <head>标签。该标签用于包含和定义 Deck 的相关信息。<head>标签之间可以包含



一个<access>标签和多个<meta>标签。

(5) <access/>标签。它的一般形式是<access domain="域" path="/路径" />, 主要用于指定当前 Deck 的访问控制信息, 有两个可选的属性。其中, domain 用来指定域, 默认值为当前域, path 用来指定路径, 默认值为 “/”, 即根目录。由于<access>单独使用, 所以要用 “/” 结尾, 后面我们还会系统地讲解 WML 的各种标签, 这里即使看不懂也没关系, 只要有些感性认识就可以了。

(6) <meta...>标签。它的一般形式是<meta 属性 content="值" scheme="格式" forua="true|false"/>, 用于提供当前 Deck 的 meta 信息, 包括内存数据处理方式, 以及数据传输方式和处理方式等。有关该标签的详细内容我们后面会专门给出。

(7) <card>标签。一个 Deck 可以包含多个 Card, 每个 Card 的内容可能不止一屏显示。对于每一个 Card, WML 均使用<card>和</card>进行包含和定义。<card>同时可以包含多个可选的属性, 如 <card id="name" title="label" newcontext="false" ordered="true" onenterforward="url" onenterbackward="url" ontimer="url">。至于这些属性的具体含义及功能, 我们将在后面介绍。

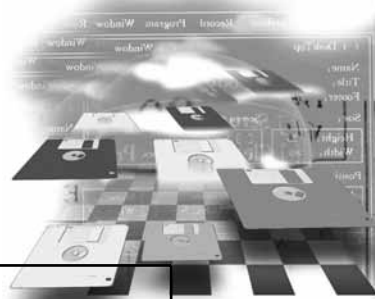
5.3 WML 语言的基本知识

上一节我们介绍了 WML 程序的基本结构, 接下来我们介绍 WML 语言的基本知识, 主要包括 WML 的字符集、变量、数据类型及 WML 程序的基本组成部分等。

5.3.1 WML 的字符集及编码

WML 使用 XML 的字符集, 即通用字符集 ISO/IEC-10646, 也即统一字符编码标准 Unicode 2.0。同时, WML 还支持其他系列的字符集子集, 例如 UTF-8、ISO-8859-1 或 UCS-2 等。其中:

UTF-8 是指通用字符集 UCS(Universal Character Set)的转换格式 8(Transformation Format 8), 主要用作传输国际字符集的转换编码。UTF-8 采用了 UCS 字符的 8 位编码, 提供了十分安全的编码格式, 可以有效避免数据传输过程中的窃听、截取及非法解密。同时, UTF-8 与 7 位 ASCII 码完全兼容, 不会影响此类编码实现的程序; 它的编码规则十分严格,



能够有效避免同步传输错误，而且还为支持其他字符集提供了足够的空间。

ISO-8859-1 字符集是国际标准化组织 ISO(International Standardization Organization)制定的 ASCII 字符集的扩展集，能够表示所有西欧语言的字符。与 ISO Latin-1 一样，ISO-8859-1 与 Windows 环境中普遍使用的美国国家标准协会 ANSI(American National Standards Institute)的字符集极为类似，绝大多数情况下无需区分。在不特别指明的情况下，HTTP 协议均使用 ISO Latin-1 字符集。因此，为了在 WML 页面中表示非 ASCII(non-ASCII)字符，开发人员需要使用相应的 ISO Latin-1 编码的字符。

UCS-2 是 ISO 10646 标准中定义的通用多 8 位编码字符集(Universal Multiple-Octet Coded Character Set)的 2 字节(即 16 位)编码标准，其字符编码值与 Unicode 字符的标准编码值相等。

WML 文档可以采用 HTML 4.0 规范所定义的任何字符编码标准进行编码处理。一般说来，WML 文档的字符编码时需要转换为另外的编码格式，以与 WAP 用户的手机浏览器所用字符标准相适应，否则，手机浏览器就无法显示 WML 页面中的字符。然而，编码转换时可能会丢失一些字符信息，所以，如果在用户端进行 WML 文档的编码转换，那么就可能导致某些结果信息丢失而不能被用户所浏览。因此，如有必要，我们应当尽量在 WML 页面传送到用户浏览器之前完成编码转换。

为了解决这一问题，一方面，我们需要为 Web 服务器补充定义 WML 的数据类型，以让服务器可以准确传输这些数据，另一方面，我们需要制定编码转换的原则。

上一章我们曾经讲过，设置 Web 服务器时也增加 WML 的数据类型，即“内容类型(Content-type)”，它是通过对 Web 服务器的 MIME 设置进行配置的。我们补充定义的 WML 及 WMLScript 数据类型共有 8 种，其中前 4 种为必选，后 4 种可根据需要选用：

wml: text/vnd.wap.wml

wmlc: application/vnd.wap.wmlc(经过编码 WML 的数据类型)

wmls: text/vnd.wap.wmlscript

wbmp: image/vnd.wap.wbmp (BMP 图像)

wmlsc: application/vnd.wap.wmlscriptc

wmlscript: text/vnd.wap.wmlscript

ws: text/vnd.wap.wmlscript

wsc: application/vnd.wap.wmlscriptc

增加上述数据类型后，我们还要规定用户浏览器对 WML 文档进行字符编码处理的一般原则：

- 其一，根据 Content-Type 传输报头的 charset 参数，如 WSP、HTTP 等；
- 其二，根据文档内嵌的 meta 信息，如 meta 元素 http-equiv 中的 charset 参数等；
- 其三，根据 XML 规定的编码；
- 其四，根据用户设置或其他编码方式。

这样，我们就可以给出 WML 对字符编码的参考处理模型。WAP 用户的手机浏览器必须执行该处理模型，或与之完全类似的模型，即：用户浏览器所能识别的任何字符编码必须与 Unicode 的所有字符正确地映射，且任何实体的处理都要在文档字符集内完成。

5.3.2 WML 字符使用基本规则

WML 是一种比较严格的语言，字符使用必须遵守相应的规则，这些基本规则主要包括以下几个方面：

(1) 大小写敏感。在 WML 中，无论是标签元素还是属性内容都是大小写敏感的，这一点继承了 XML 的严格特性，任何大小写错误都可能导致访问错误。

一般来说，WML 的所有标签、属性、规定和枚举及它们的可接收值必须小写，Card 的名字和变量可大写或小写，但它是区分大小写的。包括参数的名字和参数的数值都是大小写敏感的，例如 variable1、Variable1 和 vaRiable1 都是不同的参数。

(2) 空格。对于连续的空字符，程序运行时只显示一个空格。属性名、等号(=)和值之间不能有空格。

(3) 标签。标签内属性的值必须使用双引号(")或单引号(')括起来。对于不成对出现的标签，必须在大于号(>)前加上顺斜杠(/)，比如换行标签
必须写成
才正确。

(4) 不显示的内容。在 WML 中，不显示的字符主要包括换行符、回车符、空格和水平制表符，它们的 8 位十六进制内码分别是 10、13、32 及 9。


程序执行时，WML 将忽略所有的多于一个以上的不显示字符，即 WML 会把一个或多个连续的换行、回车、水平制表符及空格转换成一个空格。

(5) 保留字符。这是 WML 的一些特殊字符，如小于号(<)、大于号(>)、单引号(')、双引号(")、和号(&)。如果需要在文本中显示这些字符，则在程序中必须按照表 5.1 给出的方

式进行指定。这种指定方式在 WML 中称为字符的实体(Entity)，比如“&”就是“&”的实体，“<”就是小于号(<)的实体，等等。

表5.1 保留字符的指定方式

字 符	指定方式
<	< 或 <
>	> 或 >
'	' 或 '
"	" 或 "
&	& 或 & 或 #38;
\$	\$\$ 或 <
连续空格(Non breaking space)	 或
自动连字符(Soft hyphen)	­ 或 ­

 注意，表 5.1 指定方式中的分号以及普通字符标签中的分号，都是其组成部分，不能省略。如果省略了，则可能会造成 WML 编译器错误。

例如，在下面的例程中，我们就通过“<”指定了小于(<)号。当然，它也可以使用“<”来指定：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="Card_1">
    <p>
      Numerically 5 &#60; 10
    </p>
  </card>
</wml>
```

运行该程序，它在 WAP 手机浏览器上的显示结果如图 5.3 所示。



 注意，采用图 5.3 所示手机模拟图来表现程序运行结果的形式虽然直观，但需占用较多的篇幅。为了节省篇幅，我们以后只把其中浏览器屏幕截取下来说明问题。这一点希望读者能够谅解。



图 5.3 显示小于号(<)

(6) 显示汉字。如果希望 WML 程序执行时能够显示汉字，则

只需在程序开头使用 `encoding` 指定汉字字符集即可。例如：`<?xml version="1.0" encoding="gb2312"?>`。

 注意，指定汉字字符集的形式和方法可能因开发工具或 WAP 手机的不同而不同，上述方法已在 Motorola L2000www 上测试通过。其他环境下的设置方式，请读者参考开发工具使用说明或 WAP 手机使用说明。

5.3.3 变量

WML 编程中可以使用变量，变量使用前必须进行定义。变量一旦在 Deck 中的某一个 Card 上定义过，其他 Card 则可以不必重新定义就能直接调用该变量。

定义变量的语法格式为：

`$identifier`

`$(identifier)`

`$(identifier:conversion)`

其中 `identifier` 指变量名，或说变量标识符；`conversion` 指变量的替代。

变量名是由 US-ASCII 码、下划线和数字组成的，并且只能以 US-ASCII 码开头。变量名严格区分大小写，也即，变量名是大小写敏感的。

定义变量的语法在 WML 中享有最高的解释优先级。

有关变量的使用说明如下：

- (1) 在 WML 中，变量可以在字符串中使用，并且在运行中可以更新变量的值。
- (3) 当变量等同于空字符串时，变量将处于未设置状态，也就是空(Null)。
- (4) 当变量不等于空字符串时，变量将处于设置状态，也就是非空(Not Null)状态。
- (4) 在“\$identifier”形式下，WML 通常以变量名后面的一个空格表示该变量名的结束。

如果在某些情况下空格无法表示一个变量名的结束，或者变量名中包含有空格，则必须使用括号将变量名括起来，即采用“\$(identifier)”的形式。

WML 程序中的变量是可以替代的，我们可以把变量的数值赋给 Card 中的某一文本。有关变量替代说明如下：

- (1) 在 WML 程序中，只有文本部分才可以实现替代。
- (2) 替代一般在运行期发生，而且替代不会影响变量现在的值。



(3) 任何标签和属性都不能使用变量来替代。

(4) 替代是按照字符串替代的方式实现的。

(5) 如果一个没有定义的变量要实现替代, 那么该变量将被看作空字符串对待。

由于变量在语法中有最高的优先级, 包含变量声明字符的字符串将被当作变量对待, 所以如果要使程序显示“\$”符号, 则需要连续使用两个“\$”进行说明。例如: `<p> Your account has $15.00 in it</p>` 一句的显示结果为: Your account has \$15.00 in it。

5.3.4 WML 核心数据类型

WML 的核心数据类型均属于字符型数据, 是根据 XML 的数据类型定义的, 共有下述 4 种类型:

(1) CDATA 型。这种数据类型是 WML 用得最多的一种, 可以是数字、字符串或包含数字的字符串。不过定义时, 不论是数字或字符串, 都必须以文本的形式定义, 即数据用引号引起来。CDATA 型的数据仅用于属性值。例如 `$(value)` 或 `name="value"` 等。注意, 这里的 value 指 CDATA 型的数据值。

(2) PCDATA 型。这是从 CDATA 中分解出来的一类数据, 除了可以是文本形式的数字、字符串或两者的混合串外, 还可以是 WML 的标签。PCDATA 型的数据只能用于 WML 的元素表示。例如, `Text written <big> IN CAPS </big>` 一句中的 `<big>` 与 `</big>` 标签即属于 PCDATA 型的数据。

(3) NMTOKEN 型。这是一类特殊的数据, 凡是包含或部分包含数字、字母及标点符号的数据均属于 NMTOKEN 型数据。这种数据可以用标点符号开头, 但不用于定义变量名或元素名。例如, 下述 4 个数据均属于 NMTOKEN 型:

`"text"`

`_card1`

`a.name.token`

`.a-perfectly-valid.name.token`

(4) id 型。专门用于定义 WML 元素名称的数据类型。例如 `<card id="card1"/>` 中就使用 id 指定当前卡片的名称为 card1。

在这 4 种类型中, CDATA 型用起来比较灵活, 它可以使变量或数据免于语法检查。这是因为, CDATA 内的数据内容都会被当作文本来处理, 从而可以避开 WML 的语法检查,

直接作为文本显示出来。具体用法是，以“<![CDATA[”开头，“]]>”结尾，中间包含的数据均按 CDATA 型处理。例如：

```
<![CDATA[ this is <b> a test ] ]>
```

其显示结果为：this is a test。

5.3.5 WML 数据值性质

除了 NMTOKEN 型数据外，WML 其他 3 种类型的数据都必须以文本形式即加上引号进行定义。我们关心的是，这些类型的数据可以表示哪些数据值呢？或者说，它们所表示的数据值的性质是什么呢？


事实上，WML 数据值在性质上可以是长度(Length)、宏变量(Vdata)、流(Flow)、内行(Inline)、布局(Layout)、文本(Text)、超链(Href)、布尔值(Boolean)、数值(Number)或增强方式(Emphasis)，下面我们就介绍一下它们的规则及使用方法。

(1) 长度(Length)。WML 编程中使用的长度主要用于表示屏幕像素的多少或所占屏幕水平、垂直空间的百分比。比如，“50”意为 50 个像素，“50%”意为屏幕水平或垂直空间长度的 50%。长度属于 CDATA 型数据，常用“%length”表示。

例如，下例中的 hspace 与 vspace 都属于 %length 性质的数据：

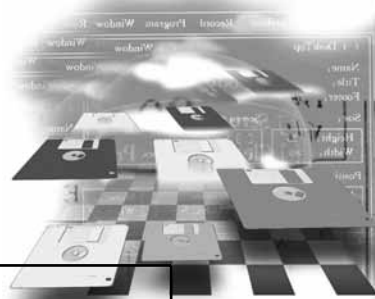
```


```

 **注意**，本章给出的例程都或多或少地用到了 WML 的标签，这部分内容我们下一章介绍。如果在这里读者一时看不懂，可以先浏览下一章的内容，稍有感性认识后再学习这部分内容。

(2) 宏变量(Vdata)。主要用于表示包含变量名的字符串，仅用于属性值。它属于 CDATA 型数据，常用“%vdata”表示。例如，<card id="card1" title="\$ (showme)">一句中的“\$ (showme)”就属于 %vdata 性质的数据。

(3) 文本(Text)。用于表示包含一定格式的文本实体，属于 PCDATA 型数据，常用“%text”表示。例如，下例中包含有运行时显示为普通格式的文本，以及加黑显示的文本：



A line with plain text.

```
<em>
```

```
  <strong>
```

A line with strong emphasis.

```
  </strong>
```

```
</em>
```

(4) 流(Flow)、内行(Inline)和布局(Layout)。流(Flow)用于表示“卡片级(card-level)”的信息，内行(Inline)用于表示“文本级(text-level)”的信息，布局(Layout)用于表示与文本布局有关的信息，如换行符等。

一般来说，内行型(%inline)的可能是文本型(%text)，也可能是布局型(%layout)的，通常是程序中的纯文本或处理的变量；而流型(%flow)的则可能包括内行型(%inline)、图像(img)、锚(anchor)及表格(table)等卡片级的所有元素。有关“锚(anchor)”的概念我们后面会讲解的。比如，下面的例程中就有换行符、文本等性质的内容，也即数据。

```
<em>
```

An emphasized line.

```
  <big>
```

A big and emphasized line.

```
  </big>
```

```
</em>
```

A line with no text formatting.

(5) 超链(Href)。主要用于表示相对的或绝对的统一资源定位符 URI(Uniform Resource Identifier)，以及 URL 地址、文件名、文件路径等。超链(Href)属于宏变量，即%vdata 型性质的数据，常用“%href”表示。例如，下述例程中的超链接、文件路径及文件名、卡片名、图像的 SRC 属性文件名等均属于%href 性质的数据：

```
<go href="http://wapforum.org/" />
```

```
<go href="file:///d:/dir/file.wml" />
```

```
<go href="app.wml" />
```

```
<card onenterforward="#card2" />
```

```

```

(6) 布尔值(Boolean)。用于表示“真(true)”或“假(false)”的逻辑值，常用“%boolean”

表示。下例程序中就使用了%boolean 型性质的数据:

```
<card newcontext="true"/>
<do optional="true" type="accept"/>
```

(7) 数值(Number)。用于表示大于或等于零的整数,常用%number 表示,属于 NMTOKEN 型的数据。例如,下例中的整数均属于%number 型性质的数据:

```
<select tabindex="2"/>
<input name="setvar" size="4" maxlength="20" tabindex="3"/>
```

(8) 增强方式(Emphasis)。这类性质的数据包含了 WML 编程中用于定义文本格式的各种标签,如、、、<i>、<u>、<big>、<small>等,常用“%emph”表示。这些标签可以定义文本以黑体、斜体、下划线等格式显示。下例中就包含了几个%emph 型性质的数据:

```
<em>
    An emphasized line.
<big>
    A big and emphasized line.
</big>
</em>
```

5.3.6 卡片与卡片组

前面我们分析 WML 程序的结构时,曾经讲到 WML 文档的信息是通过卡片(Card)集和卡片组(Deck)集的形式进行组织的。一个 Deck 是一个或多个 Card 的集合。当客户终端发出请求之后, WML 即从网络上把 Deck 发送到客户的浏览器, Deck 是服务器发送信息的最小单位。用户浏览器收到 Deck 后,可以浏览其中包含的所有 Card。Card 用于表示或描述一个或多个用户交互单位,例如, Card 可以是一个选择菜单、一屏文本或一个文本选项。逻辑上来讲,用户通过浏览器浏览时,浏览到的就是一个一个的 Card,无论是选择项目、跳转内容或阅读文本,面对的都是 Card。

图 5.4 给出了卡片组(Deck)和卡片(Card)的相互关系示意图。

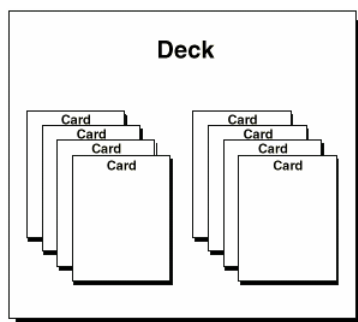


图 5.4 卡片与卡片组

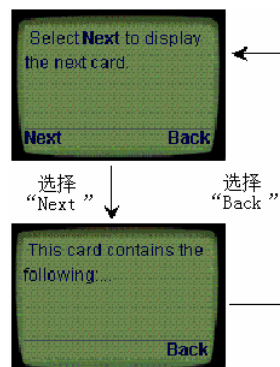


图 5.5 包含两个卡片的卡片组

下面我们给出一个实例。这个例子中有一个简单的卡片组，它包含有两个卡片。程序运行先显示图 5.5 所示的上一屏信息，即第一个卡片的内容。用户从中选择“Next”，则将显示第二个卡片的内容，即图 5.5 所示的下一屏信息。从中选择“Back”，则可返回上一卡片。程序如下：

```
<?xml version="1.0"?>                                <!-- 1 -->
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">          <!-- 2 -->
<wml>                                                  <!-- 3 -->
  <card id="First_Card">                               <!-- 4 -->
    <do type="accept" label="Next">                   <!-- 5 -->
      <go href="#Second_Card"/>                       <!-- 6 -->
    </do>                                              <!-- 7 -->
    <p>                                                 <!-- 8 -->
      Select <b>Next</b> to display the next card.      <!-- 9 -->
    </p>                                              <!-- 10 -->
  </card>                                              <!-- 11 -->

  <card id="Second_Card">                             <!-- 12 -->
    <p>                                                 <!-- 13 -->
      This card contains the following:...             <!-- 14 -->
    </p>                                              <!-- 15 -->
  </card>                                              <!-- 16 -->
</wml>                                                <!-- 17 -->
```

该程序主要语句行解释如下:

(1) 头两行语句,即注释有`<!-- 1 -->`和`<!-- 2 -->`的两行,用于声明 WML 程序。所有 WML 程序的卡片组都必须在开头部分包含这两行声明语句。

(2) 第3行用于定义WML卡片组的头。WML的卡片组必须使用`<wml>`标签开头, `</wml>`标签结尾。

(3) 第4行用于定义第一个卡片的头。与卡片组类似,每个卡片的内容也需要使用卡片标签,即`<card>`和`</card>`包含起来。卡片属于 WML 的元素,定义时需要指定一定的属性,这通常通过给标签附加属性的形式实现。第4行在定义卡片的同时还使用 `id` 属性指定了它的名称“First_Card”。

(4) 第5行定义了一个动作(Action),指定当按下功能键时,用户浏览器应该进行的操作。其中, `type` 属性定义了 WAP 手机上的 ACCEPT 功能键, `label` 属性定义了与功能键对应的标注信息,也即选项名 Next。

(5) 第6行定义了与功能键动作相适应的具体行为。其中 `href` 属性指定了目标 URI 地址,在这里是第二个卡片,即 Second_Card。

(6) 其余各行主要用于显示文本信息,比较简单,我们不再一一展开。

5.3.7 卡片组模板

同一卡片组通常会含有许多卡片,这些卡片的定义、属性或格式等通常大同小异。如果我们逐一各个卡片,显然是十分麻烦的。为此, WML 提供了卡片组模板的功能,模板内定义了一系列标准和参数,可以应用到同一卡片组的所有卡片中去,从而能够大大地提高我们的编程效率。有关卡片组模板的内容我们后面会专门介绍。

5.3.8 WML 与 URL、程序段锚点

我们知道,环球网 WWW(World Wide Web)是各种信息和设备的网络,为保证全球范围内的交互,人们制定了3种规范:其一,统一资源定位器 URL(Uniform Resource Locators),提供所有网络资源的标准命名方式和定位方式;其二,标准协议,如 HTTP 协议等,提供 WWW 资源的传输方式;其三,标准内容类型,如 HTML、WML,提供 WWW 资源的内



容形式及标准。WML 沿用了这些规范，并扩大了 URL 使用的范围。在 WML 中，不仅超链接、文件路径及文件名可以作为 URL 处理，卡片名、宏变量名及各种内部资源名等也可作为 URL 处理。

为此，WML 改进了 HTML 命名资源位置的方式，采用程序段锚点(Fragment Anchor)的形式来处理 WML 程序中某段程序的定位。程序段锚点根据文档 URL 规则进行定义，并按照程序段标识符前加井字号(#)的方式书写。使用程序段锚点，WML 程序可以在同一卡片组中定位不同的卡片。如果在程序中不指定程序段，那么程序中引用的 URL 名称则指整个卡片组，而且卡片组的名称同时也是本卡片组内的第一个卡片的名称。

例如，`<go href="#Next_Card"/>`一句中的 go 元素就包含了一个 URL 地址，该地址指定了同一卡片组中的另一个卡片。该 URL 地址就包含了程序段标识符(#)，`"#Next_Card"` 就是一个程序段锚点。

WML 还改进了相对 URL 地址的用法。通过类似于相对路径的定位方式，实现相对 URL 地址的处理。其格式为“/目录名/子目录名/.../文件名”，例如 `“/options/foo.wml”` 就是一个相对 URL 地址。

下面的实例就说明了相对 URL 地址的用法。当用户选择执行 go 任务后，用户的浏览器就会定位并执行 `“/options/foo.wml”` 指定的网页：

```
<wml>
  <card>
    <do type="options" label="Options">
      <go href="/options/foo.wml"/>
      label="menu"
    </do>
    <!-- rest of the card -->
  </card>
</wml>
```



5.3.9 浏览器操作历史

为了在浏览器端管理 WML 程序的执行, WML 使用“浏览器前后关系(browser context)”的功能来保存 WML 程序执行的状态及各种参数、变量等, 这样可以记录浏览器端用户的操作情况。同时, WML 还提供了一个简单的导航历史模型, 以 URL 地址的形式记录了用户浏览时的各种操作, 并把这些 URL 地址放入历史堆栈。通过该堆栈, 用户即可实现历史浏览的回溯及其他操作。

目前, WML 在用户浏览器端提供了 3 种针对历史堆栈的操作: 重置、推进和取出。

(1) “重置(Reset)”操作可以将历史堆栈保存的信息恢复到当前卡片的最初状态信息。

(2) “推进(Push)”操作可以把用户浏览的新卡片的信息作为一个新的 URL 地址, 保存进历史堆栈中, 以备后用。

(3) “取出(Pop)”操作则可在用户回溯以前浏览的卡片时, 把该卡片的状态信息从堆栈中取出来, 并应用到卡片显示中去。

本章小结

本章先在介绍 WML 支持的 WAP 设备特点的基础上, 通过实例讲解了 WML 程序的编辑和测试方法。然后, 通过大量 WML 小程序, 分析了 WML 程序的结构及一般组成, 讲解了 WML 语言的基本知识, 包括 WML 的元素、标签、字符集、编码方式、变量、数据类型、卡片、卡片组等。本章内容比较零碎, 涉及的基本概念比较多, 这些概念对读者理解 WML 程序的特点及掌握 WML 编程的基本思路十分重要, 希望大家能够认真学习这部分内容。



第 6 章 WML 编程

元素和标签是 WML 的主要语法，它们决定了 WML 编程的基本规则。本章我们就从 WML 的元素、标签、属性等方面详细讲解 WML 的编程方法。学习本章知识之前，读者应当了解 WML 元素与标签的区别。WML 的元素通常由一个首标签、内容、其他元素及一个尾标签组成，元素也可以不含有内容。不包含内容的元素称为空元素，它退化成一个单独的标签。也就是说，单独的标签是一个元素，成对出现的标签与其包含的内容也构成一个元素。由于元素涉及标签，标签又涉及属性，所以我们本章各节均以元素为主线，通过大量实例分别从卡片与卡片组、事件、任务、变量、用户输入、锚点、图像、时间控制、文本格式等几个方面进行讲解。

6.1 卡片、卡片组及其元素

我们前面已经多处介绍过 WML 的卡片与卡片组，主要从概念和相互关系的角度进行了分析。我们这里则从卡片、卡片组的组成、相关元素、标签及属性等编程角度进行分析和讲解。

6.1.1 共有属性

WML 元素的共有属性主要有 3 个，即 id、class 和 xml:lang 属性。

WML 的所有元素都有两个核心属性，即标识(id)和类(class)属性。它们主要用于服务器方的信息传输。其中，id 属性用于定义元素在卡片组中的唯一标识，即它的名称；class 属性用于给当前元素定义一个或更多的类(class)。与卡片组一样，类(class)也是有名字的，而且多个元素可以使用一个类(class)名。具有相同类名的单一卡片组中的所有元素均可被看作相同类的一个部分。类名是区分大小写的。如果在 class 属性列表中，一个元素具有多个唯

一的类名，那么该元素可以看作这些类中的一部分。具有同一属性的多个类名必须用两个以上的空格间隔，WML 程序执行时将忽略其中多余的类名及其属性。

另外，在 WML 程序，所有包含文本的元素均具有 “xml:lang” 属性。该属性用于指定当前元素及其属性所用的描述语言，如英国英语、美国英语、法语、德语等，并可以为用户浏览器选择显示文本的语言提供依据。

6.1.2 WML 程序的文件头

合法的 WML 卡片组均属合法的 XML 文件，因此它必须包含 XML 的声明及文件类型的声明。典型的 WML 程序的文件头包括我们前面多次提到的以下两行程序：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
```

编写 WML 程序时，我们必须写入这两行程序，并放在程序开始的第 1、2 行。其中，“-//WAPFORUM//DTD WML 1.1//EN” 是标准通用标记语言 SGML(Standard Generalized Markup Language)的公共标识；“http://www.wapforum.org/DTD/wml_1.1.xml” 是 WML 程序文档类型的标识。文档类型标识也可以是“text/vnd.wap.wml”或“application/vnd.wap.wmlc”，其中前者指定 WML 的原文类型，后者指定 WML 程序编译后代码类型。

6.1.3 wml 元素

WML 的 wml 元素用于定义一个卡片组，并通过<wml>与</wml>标签包含和封装该卡片组中的所有卡片及信息。它的语法格式如下：

```
<wml xml:lang="lang">
    内容(content)
</wml>
```

其中 xml:lang="lang"用于指定文档所用语言(前面已有介绍)，语言"lang"的值属于 NMTOKEN 型数据。

wml 元素中包含的内容(content)中除了文本、图像等信息之外,还可以包含 head、template 及 card 元素。其中 head、template 元素如果包含的话则只可包含一次,而 card 元素必须至少包含一次。有关这些元素的用法我们后面介绍。

下面给出的例程中,wml 元素中包含了一个由两个卡片构成的卡片组:

```
<wml xml:lang="en-us">
  <card id="card1" title="Card 1">
    <do type="accept">
      <go href="#card2"/>
    </do>
    <p>
      Hello world!
      This is the first card...
    </p>
  </card>

  <card id="card2" title="Card 2">
    <p>
      This is the second card.
      Goodbye.
    </p>
  </card>
</wml>
```

程序第 1 行中的 xml:lang 属性用于指定文档使用美国英语("en-us")编写。运行该程序时,它将先在 WAP 用户的手机浏览器上显示第 1 个卡片,当用户激活 do 元素后,则显示第 2 个卡片,效果如图 6.1 所示。

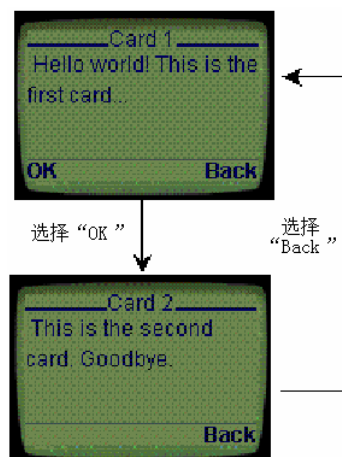


图 6.1 wml 元素应用举例

6.1.4 template 元素

template 元素用于为当前卡片组中的所有卡片定义一个模板,统一规定卡片的某些参数。模块中的事件处理功能则可将这些参数自动应用于同一卡片组中的所有卡片。不过,我们也可以使其中某个或某几个卡片不采用模板规定的形式,方法是在该卡片中定义一个同名的事件来替代模块中相应的事件。template 元素是通过<template>和</template>标签包

含所需内容(content)而实现模板功能的,其语法格式如下:


```
<template onenterforward="href" onenterbackward="href" ontimer="href" >
    内容(content)
</template>
```

template 元素包含的内容中,除了卡片的一般参数外,还可以包含任意多次的 do 元素和 onevent 元素。template 元素各属性的功能及用法说明如下:

(1) onenterforward。当用户在浏览器中进入当前卡片时,该属性将指定超链(href)的 URL 地址,浏览器将据此打开 URL 指定的卡片或事件。

(2) onenterbackward。与上一属性类似,该属性也可以指定其相应卡片或事件的 URL 地址。如果用户浏览时执行 prev 任务,那么浏览器就会定位到该属性所指定超链(href)的 URL 地址,并打开 URL 指定的卡片或事件。

(3) ontimer。当指定时间 timer 过期的时候,用户浏览器就根据 ontimer 属性指定的 URL 打开相应的卡片。

 有关 onenterforward 事件、onenterbackward 事件和 ontimer 事件的详细情况我们后面将专门介绍。

下面举出的关于 template 元素的例子中,包含有一个 do 元素,指定当用户选择 prev 按钮功能时,就打开浏览的前一个卡片:

```
<template>
    <do type="prev" label="Previous">
        <prev/>
    </do>
</template>
```

下面的例子稍微复杂一些。程序开始定义了一个模板,包含 do 元素,指定当用户从选项(options)中选择“do1”时,打开系统默认的卡片。随后,程序定义了卡片“first”,它自动套用了刚刚定义的模板。接下来,程序又定义了卡片“second”,并使用同名的事件处理替代了模板中定义的事件处理,即当用户选择“do1”时,该卡片将执行<go href="/options"/>。大家可以仔细阅读并分析一下这个程序:



```

<wml>
  <template>
    <do type="options" name="do1" label="default">
      <prev/>
    </do>
  </template>

  <card id="first">
    <!--该卡片将自动套用模板中定义的事件处理过程-->
    ...
  </card>

  <card id="second">
    <!--该卡片使用同名的事件处理替代了模板中提供的事件处理-->
    <do type="options" name="do1" label="options">
      <go href="/options"/>
    </do>
    ...
  </card>
</wml>

```

6.1.5 card 元素

WML 的卡片组是由一个或多个卡片(card)构成的，每个卡片都包含有一套用户和浏览器交互操作的配置及模式。用户对交互操作的需求是多样性的，所以卡片定义时也必须是多样性的。为此，WML 提供了 card 元素，通过<card>和</card>标签定义一个卡片的各种属性、包含内容。它的语法格式如下：

```

<card id="name" title="label" newcontext="boolean" ordered="true" onenterforward="href"
  onenterbackward="href" ontimer="href">
  内容(content)
</card>

```

card 元素中包含的内容(content)中除了文本、图像等信息之外，还可以包含 onevent、

timer、do 和 p 元素。其中, timer 元素只可使用一次, 其余 3 种可使用多次。而且, 如果 card 元素包含 onevent 元素或 timer 元素的话, 那么 onevent 元素必须放在最前面, timer 元素放在 onevent 元素的后面, 随后才可以使用 do 或 p 元素。这个优先顺序是不能乱的。

card 元素各属性的功能及用法介绍如下:

(1) id。用于指定 card 的名字。该名字是程序导航定位的依据, 可以用作程序段锚点, 比如 `<go href="#cardname"/>`。其中的 cardname 便是由 id 指定的卡片名。

(2) title。用于为卡片指定一个简短的标题或说明信息。

(3) newcontext。用于指定 WAP 手机浏览器当用户重新进入的时候是否需要初始化卡片中所有的内容。它有 true 和 false 两种选择, 当指定 newcontext="true" 时, 卡片的所有内容在用户重新进入时将进行初始化, 并清除历史记录; 否则, 指定 newcontext="false" 时, 将不进行初始化, 也不清除历史记录。默认状态下的设置值为 false。另外, newcontext 仅当作为 go 任务的一部分时才可被执行。

(4) ordered。用于向用户手机的浏览器指明卡片内容的组织形式, 以便让浏览器根据自身特点及卡片内容组织及时安排显示布局。它有两种布尔值的设置, 即 true 和 false。

当 ordered="true" 时, 浏览器将按照线性顺序显示卡片各区域的内容。这个线性顺序通常是大多数用户所习惯采用的信息浏览顺序, 比如发送 E-mail 信息时, 我们依次需要 E-mail 收件人地址、主题及 E-mail 内容, 这个逻辑顺序就属线性顺序。

当 ordered="false" 时, 浏览器将根据用户选择或指定的顺序来显示内容。这种情况主要适用于显示用户选项、无序组件或用户输入的简单数据记录等。

(5) onenterforward。onenterforward 事件仅当用户使用 go 任务或类似于 go 的任务定位和浏览卡片时才可发生, 即如果用户执行 go 任务, 则浏览器就会定位 `<go>` 标签中指定超链(href)的 URL 地址, 并打开 URL 指定的卡片。card 元素中的 onenterforward 属性是 onevent 元素的一个简短格式, 用于直接指定 onenterforward 事件的 URL 地址。

(6) onenterbackward。该属性可以指定其相应事件的 URL 地址。如果用户浏览时执行 prev 任务, 那么浏览器就会定位到该属性所指定超链(href)的 URL 地址, 并打开 URL 指定的卡片。onenterbackward 属性也属于 onevent 元素的一个简短格式。

(7) ontimer。当指定时间 timer 过期的时候, 用户浏览器就根据 ontimer 属性指定的 URL 打开相应的卡片。它也属于 onevent 元素的一个简短格式。

下面我们给出一个内含卡片的 WML 卡片组的例子, 其中由 card 元素实现的卡片包含



有标题、简单的文本等信息：

```
<wml>
  <template>
    <do type="accept" name="exit" label="EXIT">
      <prev/>
    </do>
  </template>

  <card id="card_1" title="Welcome" newcontext="true">
    <p>
      Hello World!
    </p>
  </card>
</wml>
```



图 6.2 card 元素应用举例

该程序的运行效果如图 6.2 所示。

6.1.6 head 元素

head 元素用于指定卡片组的头，即与卡片组整体有关的信息，包括 meta 数据和 access 控制信息。它通过<head>和</head>两个标签进行定义，其语法格式如下：

```
<head>
  内容(content)
</head>
```

head 元素包含的内容(content)中，应至少有一次 meta 元素或 access 元素。

有关 meta 元素及 access 元素的用法我们随后介绍。head 元素的举例可参见后面 meta 元素及 access 元素的举例。

6.1.7 access 元素

access 元素是由一个单独的标签即<access>标签实现的元素，用于定义 WML 整个卡片组的操作权限，即访问控制参数。access 元素必须在 head 元素内和其他的 meta 信息一起声

明，而且每个卡片组只能有一个 access 元素。其语法格式如下：

```
<head>
<access domain="domain" path="path"/>
...
</head>
```

access 元素属性的功能及用法如下：

(1) domain。用于指定可对卡片组进行操作的 URL 域，默认域是当前卡片组所在的域。domain 的目的是限制访问，用户浏览时浏览器将根据 domain 值所规定的每个子部分的值来得出与之匹配的地址，并访问该地址对应的卡片或事件。比如，<access domain="mywapnet.com"/>规定了 access 的域值为 mywapnet.com，所以，用户链接到“http://www.mywapnet.com/”时将是有效的，而链接到“http://www.mywap.com/”或“http://www.mywapnet.net/”等地址时则是无效的。

(2) path。用于指定卡片组操作的其他卡片组所在的根目录。默认目录是“/”，即当前卡片组所在的根目录。默认目录的规定使得所有在 domain 域下的卡片组都可以操作当前卡片组。path 的值是访问时需要匹配的路径，它的工作原理与 domain 十分相似，需要与路径的每个子路径相匹配，否则均属无效。比如，<access path="/internal"/>指定了 access 的路径值为“/internal”时，则只有路径“/internal/”是有效的、符合匹配要求的，诸如“/internal-wml”、“inter/”的其他路径则是无效的。

举例来说，<access domain="acmecorp.com" path="/pub"/>一句定义了域为“acmecorp.com”，路径为“/pub”，所以，用户通过浏览器访问下述 URL 地址指定的卡片组就是有效的、合法的：acmecorp.com/pub/stocks.cgi 或 www.acmecorp.com/pub/demos/packages.cgi；而不能访问下述 URL 地址的卡片组：www.test.net/pub 或 www.acmecorp.com/internal/foo.wml。

6.1.8 meta 元素

meta 元素用于定义 WML 卡片组相关的通用 meta 信息。该元素是由一个单独的标签即 <meta/> 标签实现的元素，其语法格式如下：


```
<meta name="name"|http-equiv="name" content="value" forua="true|false" scheme="format"/>
```

其中, name 属性和 http-equiv 属性只能选择使用一个; content 属性是必选的, 其值根据属性而定; scheme 属性目前尚不支持; forua 属性为可选属性。各属性功能及用法说明如下:

(1) content。该属性用于指定 meta 信息的性质的值, 是必选的。

(2) name。用于指定 meta 信息性质的名称。用户浏览器通常忽略已经命名的 meta 数据, 网络服务器也拒绝发送包含该属性所指定 meta 数据名称的内容。

(3) http-equiv。该属性用于替代 name 属性, 可将 meta 数据转为 WSP 或 HTTP 协议的响应头。

(4) forua。该属性用于指定那些开发者希望传送至用户浏览器的性质。它有 true 和 false 两个取值, 如果取 false, 则卡片组在发送往客户端以前必须由中间代理去除 meta 元素信息, 这是因为传输的协议可能改变; 若取值为 true, 则 meta 数据必须如实送往用户的浏览器。默认状态下, 该属性的值为 false。

(5) scheme。该属性用于指定解释 meta 信息性质值的形式或结构, 具体的形式或结构因 meta 数据的类型不同而不同。

我们举例来看几种 meta 数据的定义:

`<meta http-equiv="Cache-Control" content="max-age=3600"/>`可以指定卡片组在手机内存缓冲区中的存储时间段, 内存不耗尽的条件下默认时间段为 30 天。在此期间内, 手机能够直接从缓存里调用访问过的卡片组。如果信息可以设置时间, 则可用 max-age 指定卡片组在缓存里的生存期, 最小单位是秒; 若指定为 0, 则每次都需通过连接服务器来调用该卡片组。

`<meta user-agent="vnd.up.bookmark" content="指定的 URL"/>`和 `<meta user-agent="vnd.up.markable" content="false"/>`可以实现类似于普通浏览器的书签功能。当用户给某一个卡片做了书签之后, 手机浏览器将首先使用一个标记记录该卡片, 该标记在默认情况下是 `<card>` 标签中 title 属性定义的标题。以后, 当用户选择了该书签以后, 浏览器就会打开被记录的 URL 地址所对应的卡片。由于在默认情况下, 手机会记录所有的卡片组, 所以 meta 元素一般需指定手机不记录当前 URL, 即 `<meta user-agent="vnd.up.markable" content="false"/>`。此外, 如果要为书签指定不同于当前卡片组的 URL, 则可用 `<meta user-agent="vnd.up.bookmark" content="指定的 URL"/>` 来实现。

下面我们再给出一个利用 meta 元素给用户浏览器指定 WML 卡片组所用字符集的简单例子，它是利用 user agent="character-set=UTF-8"实现的：

```
<head>
<access domain="mycompany.com" path="/WML" >
<meta content="charset" user agent="character-set=UTF-8"/>
</head>
```

6.2 任务及其元素

WML 允许我们在程序中指定一些任务，当某些特定的事件激活时，即可执行这些任务，从而完成需要的操作。例如，我们可以设定任务，当用户按下相应的功能键时，浏览器就可打开指定的卡片组或卡片。目前，WML 提供了 4 个任务元素，即 go、prev、noop 和 refresh，它们主要与 do 元素和 onevent 元素中指定的事件相响应。本节我们就对任务的这些元素作一详细介绍。

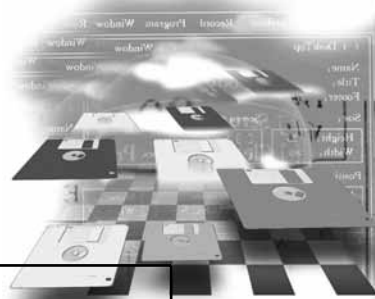
6.2.1 go 任务

go 任务是通过 go 元素来声明的，而 go 元素是通过<go>和<go/>标签进行定义的。go 元素主要用来定义浏览器需要导航的 URL 地址。如果该地址是一个 WML 卡片或卡片组的名字，则浏览器就会打开并显示相应的卡片、卡片组；否则，浏览器就会执行该 URL 指定的任务或事件等。在历史堆栈中，go 任务执行的是一个“推进(push)”操作，也就是说，它执行时浏览器浏览的 URL 地址将送入历史记录列表中，以备它用。

go 元素中可以包含任意次的 setvar 元素或 postfield 元素。postfield 元素前面已有介绍，这里不再重述，setvar 元素我们后面介绍。

go 任务的语法格式如下：

```
<go href="href" sendreferer="false|true" method="get|post" accept-charset="charset">
    内容(content)
</go>
```



其中属性的功能及用法介绍如下：

(1) href。该属性用于指定目标 URL 地址，比如让浏览器显示的卡片地址及名称等。该属性是必选的，其他属性为可选。

(2) sendreferer。该属性用于指定是否传递调用 href 所指定的 URL 的卡片的 URL，也就是当前页的 URL，即 HTTP 头中的 HTTP_REFERER。有两种选择：true 或 false。其中默认值为 false。

(3) method。与 HTML 中的表单 FORM 的 method 属性一样，该属性用于指定表单是以 get 的方式还是 post 的方式递交，以便为通用网关接口 CGI 处理。默认值为 get，但如果没有指定 method 属性，而 <go> 和 </go> 之间存在 postfield 元素，则 WAP 手机浏览器会自动以 post 方式传递。

(4) accept-charset。当 Web 服务器处理来自浏览器的输入信息时，该属性可指定服务器进行数据编码时必须采用的字符集列表。也就是说，该属性指定的字符集将替代 HTTP 头里指定的字符集，以便作为服务器选用字符集的标准。比如 accept-charset="UTF-8,US-ASCII,ISO-8859-1" 指定了 3 种字符集。

go 任务包含的内容(content)里面通常含有 postfield 和 setvar 元素，它们的应用格式一般为 <setvar name="name" value="value"/> 及 <postfield name="name" value="value"/>。

其中，setvar 元素用于指定当触发某一事件时，给变量赋值。而 postfield 元素指定了处理用户请求时，浏览器向源服务器(origin server)传输的信息，name 和 value 属性都是必选的。

在下面的举例中，go 元素为 URL 地址 "/foo?x=1" 定义了一个 HTTP GET 的请求：

```
<go href="/foo">
    <postfield name="x" value="1"/>
</go>
```

而下面的例子则为含有消息实体 "w=12&y=test" 的 URL 地址 "/bar" 定义了一个 HTTP POST 的传输方式：

```
<go href="/bar" method="post">
    <postfield name="w" value="12"/>
    <postfield name="y" value="test"/>
</go>
```

我们再看一个稍微复杂一点儿的例子。这个例子建立了一个用户交互的小部件，即一个“Help”选项，当用户选择该项，程序将被 go 元素定位并打开 help 卡片。程序如下：

```
<card id="card1">
  <do type="help" label="Help">
    <go href="#help"/>
  </do>
</card>

<card id="help">
  <p>
    Help topics:
    <!--其他帮助信息，此略-->
    ...
  </p>
</card>
```

注意上述程序中 go 元素标签的形式：`<go href="#help"/>`。一般来说，当`<go>`和`</go>`之间没有任何语句时，要采用`<go/>`的形式，也就是说，这时 go 元素“退化”为一个单独的标签。再如，下面几行程序中也包含了一个这样的 go 元素：

```
<anchor title="Link1">
  <go href="test.wml"/>
  News
</anchor>
```

6.2.2 prev 任务

prev 任务是由 prev 元素实现的。该元素通常是一个单独的标签即`<prev/>`，不过有时也可由`<prev>`和`</prev>`一对标签进行定义。它用于指定将浏览器导航至历史堆栈中的前一个 URL 地址。在浏览器操作的历史堆栈中，prev 任务执行的是“取出(pop)”操作(参见 5.3.9 节)，将前一 URL 地址取出，并把当前 URL 地址推进历史堆栈。如果历史堆栈中没有前一个 URL 地址，则 prev 元素不执行任何任务。



prev 任务的语法格式为：

```
<prev/>  
或 <prev> 内容(content) </prev>
```

在上一语法格式中，prev 元素包含的内容里面一般是 setvar 元素，该元素的含义前面已有介绍，这里不再重述，具体用法随后介绍。

我们来看一个例子，程序如下：

```
<do type="accept" label="Back">  
    <prev/>  
</do>  
  
<p>  
    Hello, World!  
</p>  
.....
```

该程序建立了一个用户交互的小部件，即在浏览器屏幕上显示一个名为“Back”的选项，当用户选择该项时，浏览器屏幕将打开前面刚刚打开过的一个卡片。

6.2.3 refresh 任务

refresh 任务由 refresh 元素声明，它用于刷新当前的卡片，对卡片内指定的变量进行更新。其语法格式为：

```
<refresh>  
    内容(content)  
</refresh>。
```

其中包含的内容(content)中一般都有 setvar 元素，其语句形式为<setvar name="name" value="value"/>，它可指定更新的变量名 name，及更新的变量值 value。另外，refresh 元素也可以不包含 setvar 元素，而通过时间限制(如 timer 元素)对卡片进行刷新。

作为举例，我们给出一个使用 refresh 元素的例子。它可以为变量 product 设置值为 foo，

并就此以新的变量值刷新浏览器的显示内容。程序如下：

```
<refresh>
  <setvar name="product" value="foo"/>
</refresh>
```

6.2.4 noop 任务

noop 任务由 noop 元素进行声明，表示什么也不做，是一个空操作，在替代卡片组一级的 do 元素时十分有用。该元素是一个单独的标签，即<noop/>标签。其语法格式如下：

```
<noop/>
```

noop 元素没有属性，下面的简单程序中就包含了 noop 元素实现的空任务操作：

```
<card id="card1">
  <do type="options" name="dome">
    <noop/>
  </do>
  ...
</card>
```

6.3 事件及其元素

WML 提供了几个元素，专门用于处理用户浏览器端的导航和事件。利用这些元素，用户可以给某任务指定关联事件。那么当事件触发时，浏览器就会执行相应的任务，比如 URL 导航就是通过事件实现的。而且，事件可以和不只一个需要完成的任务捆绑在一起。事件捆绑时一般是通过几种元素及其标签声明来实现，如 go、do 和 onevent 等元素。下面我们就讲解 WML 的事件元素及事件。

6.3.1 do 元素

do 元素提供了一个通用的事件处理机制，使得用户可以参与当前卡片的事件处理。它



通过<do>和</do>标签将用户交互和某一个任务联系在一起。用户交互可以是用户按下的功能键、选择的菜单项，也可以是用户的声音提示。当用户激活这些交互功能时，用户浏览器就会执行与 do 元素相关联的任务。其语法格式如下：

```
<do type="type" label="label" name="name" optional="boolean">  
    任务(task)  
</do>
```

其中 task 是与 do 元素关联的动作，也是条件激活时浏览器即将执行的内容。在 do 元素中，用户必须绑定且只能绑定 go、prev、noop 或 refresh 四种元素所实现任务中的一个任务(task)。go 元素用于定位指定的 URL 地址，prev 元素用于定位并打开前一操作或任务，noop 为空操作，refresh 用于刷新当前卡片组或任务，有关它们的详细用法我们后面会陆续介绍。

do 元素可以用于卡片一级，也可用于卡片组一级。当用于卡片一级时，do 元素必须包含在 card 元素中；而用于卡片组一级时，do 元素必须包含在 template 元素中，由此定义的 do 元素将同时应用于当前卡片组的所有卡片，此时如果某个卡片不想应用模板中的 do 元素及其任务，则需采用我们前面介绍的方法，使用同名事件处理来替代模板中 do 元素的事件处理。而且，不论事件关联的任务是否相同，当 do 元素定义的事件名称相同时，卡片一组的 do 元素将忽略卡片组一级 do 元素的影响，即卡片一组的 do 元素将被优先执行。

另外，含有空操作(noop)任务的 do 元素，不论它是否被激活，它都不会传送或显示到用户的浏览器中，这在一定程度上可以加快浏览器的工作效率，因为服务器端替它抛弃了一些空任务的判断。

do 元素各个属性的功能及用法讲解如下：

(1) type。用于指定 do 元素的类型(type)，也即需要关联、绑定的用户交互事件，是必选属性。用户浏览器接到这些事件后，就会激活它们并执行相应的操作与处理。如果在一个卡片中定义了多个 do 元素并拥有同样 type，那么用户必须为每个 do 指定不同的事件名才行，否则就会发生判断混乱的错误。

do 元素典型的类型(type)及执行条件介绍如下：

accept。当用户选择或按下相应功能键(ACCEPT)、选项、命令或按钮时，浏览器接受或激活当前所作选择。

prev。激活 PREV 键时，浏览器将导航到历史记录中的前一个卡片。

help。激活 HELP 功能键或相应帮助按钮、命令时，浏览器显示与当前内容相关的帮助信息。

reset。激活 RESET 功能键或相应按钮、命令时，清除或重置当前卡片组或浏览器的状态。


options。激活 OPTIONS 功能键或相应按钮、命令时，浏览器显示与当前内容有关的选项或附加操作。

delete。激活 DELETE 功能键或相应按钮、命令时，删除当前项目或选择。

unknown。如果给出的类型不能为 do 元素所识别，则一律按照 unknown 型处理，相当于类型为空，即 type=""。

vnd.*，VND.* 及其他不同大小写组合 [Vv][Nn][Dd].*。这种类型定义的都是 vnd.co-type，用于激活供应商或用户浏览器自定义的某个特定功能，其中 co 为公司(company)名的缩写。

X-*与 x-*。扩展类型，目前 WML 中还没有使用。

需要说明的是，对于 type 定义的用户交互机制，有些浏览器将该交互映射成功能按键，有些则映射成上下文独立的选项或操作。具体操作时，用户应当根据交互操作的有关提示，或浏览器、WML 程序的有关说明选择功能键、按钮或命令。

(2) label。该属性指定的文本字符串可以标识用户的交互事件。例如，当把某一任务绑定在 ACCEPT 键上之后，并设置了 label 属性，比如 label="gome"，那么浏览器就会将 label 的值“gome”显示在屏幕上；如果不指定，浏览器则会显示默认的“OK”字符串。为了保证能在较小的手机屏幕上显示出来，label 的属性值最多不能超过 6 个字符。不过这可能因 WAP 手机品牌、型号不同而稍有不同，有的手机要求最多不能超过 5 个字符。而且，如果手机浏览器不支持动态标签显示，那么它就会忽略 label 属性。

(3) name。该属性用于指定 do 元素所绑定事件的名称。如果多个 do 元素指定了相同的 name，那么它们绑定的事件同属一个。如果卡片一级与卡片组一级中的 do 元素指定了相同的事件名，那么卡片一级的事件将被优先执行，卡片组一级的事件将被忽略。

WML 规定，在同一卡片或同一模板中，不得指定具有相同事件名(name)的两个或两个以上的 do 元素。

(4) optional. 指定浏览器是否忽略 do 元素及其包含的任务。有两个可选值 :true 和 false。如果值为 true ,则浏览器将忽略当前 do 元素 ,即不执行它所绑定的任务。反之 ,若值为 false ,则执行 do 元素。

```
<card id="card1">
  <do type="accept" label="Next">
    <go href="#card2"/>
  </do>
  <p>
    Select <b>Next </b> to go to the next card.
  </p>
</card>

<card id="card2">
  <p>
    This is card 2.
  </p>
</card>
```

6.3.2 ontimer 事件

ontimer 事件只能包含在 card 元素或 template 元素的标签中进行定义，其语法形式为：



```
<card id="name" title="label" newcontext="boolean" ordered="true" onenterforward="href"
onenterbackward="href" ontimer="href" >
    内容(content)
</card>
```

或：

```
<template onenterforward="href" onenterbackward="href" ontimer="href" >
    内容(content)
</template>
```

ontimer 事件只有一个属性，即 ontimer。它用于指定一个超链(href)的 URL 地址，当指定时间 timer 过期的时候，用户浏览器就会按照该超链(href)的 URL 打开相应的卡片。

下面给出的例子就包含有 ontimer 事件，指定当浏览器等待 5 秒钟后，显示 <http://wapserver/hello.wml>：

```
<card id="cardname" ontimer="http://wapserver/hello.wml" title="title">
    <timer value="50"/>
    <p>
        Hello World!
    </p>
</card>
```

6.3.3 onenterforward 事件

onenterforward 事件仅当用户使用 go 任务或类似于 go 任务的任务来定位和浏览卡片时才可发生。设置 onenterforward 事件后，当用户进入当前卡片组时，浏览器就会定位 onenterforward 属性或<go/>标签中指定超链(href)的 URL 地址，并打开 URL 指定的卡片。

onenterforward 事件需要包含在 card 元素、template 元素或 onevent 元素的标签中进行定义，其语法格式为：

```
<card id="name" title="label" newcontext="boolean" ordered="true" onenterforward="href"
onenterbackward="href" ontimer="href" >
    内容(content)
</card>
```

```
<template  onenterforward="href"  onenterbackward="href"  ontimer="href"  >
    内容(content)
</template>
```

```
<onevent type="onenterforward">
    <go href="href"/> 或其他任务(task)
</onevent>
```

```
<card onenterforward="http://wapserver/hello.wml">
    Hello World! You came back.
</card>
```

在第三种格式中，onenterforward 事件是作为 onevent 元素的一个类型值，并结合<go/> 标签指定事件激活时浏览器需要打开的卡片的 URL 地址。例如，上述程序可以写为：

```
<card>
  <onevent type="onenterforward">
    <go href=" http://wapserver/hello.wml "/>
  </onevent>
  <p>
    Hello World! You came back.
  </p>
</card>
```

121

6.3.4 onenterbackward 事件

当用户使用 prev 任务或类似的任务来导航至某一卡片时, onenterbackward 事件才可发生。换句话说,当用户从历史堆栈(参见 5.3.9 节)中选取 URL 地址,并通过浏览器打开这一地址对应的卡片时, onenterbackward 事件才能发生。

与 onenterforward 事件类似, onenterbackward 事件也需要包含在 card 元素、template 元素或 onevent 元素的标签中进行定义,其语法格式为:

```
<card id="name" title="label" newcontext="boolean" ordered="true" onenterforward="href"
    onenterbackward="href" ontimer="href" >
    内容(content)
```

```
</card>
```

或:

```
<template onenterforward="href" onenterbackward="href" ontimer="href" >
    内容(content)
```

```
</template>
```

或:

```
<onevent type="onenterbackward">
    <go href="href"/> 或其他任务(task)
</onevent>
```

前两种格式中, onenterbackward 事件是作为 card 元素或 template 元素标签中的一个属性进行定义的,该属性即为 onenterbackward,它指定了一个超链(href)的 URL 地址,当用户使用 prev 等任务向回导航至这一地址时,浏览器将据此打开 URL 指定的卡片。

后一种格式中, onenterbackward 事件是作为 onevent 元素的一个类型值,并结合 <go/> 标签指定事件激活时浏览器需要打开的卡片的 URL 地址。例如,下面的程序就是采用第三种格式的例子:

```
<card id="card1">
    <onevent type="onenterbackward">
        <go href="#card2"/>
    </onevent>
</card>
```



```
        Hello World!
    </p>
</card>

<card id="card2">
    <p>
        You came back!
    </p>
</card>
```

在这一例子中，当用户在当前卡片即 card1 中，使用 prev 任务或历史堆栈中的向回导航功能时，浏览器就会激活 onenterbackward 事件，并导航至<go/>标签中指定 URL 地址的卡片，也就是 card2 卡片。这样，card2 的内容就会取代 card1 的内容显示在浏览器中。当然，如果此时用户使用 go 任务向前导航，则又会打开 card1 卡片。

上述程序也可以采用 card 元素或 template 元素的格式来实现，例如，使用 card 元素实现时，程序如下：

```
<card id="card1" onenterforward="#card2">
    <p>
        Hello World!
    </p>
</card>

<card id="card2">
    <p>
        You came back!
    </p>
</card>
```

6.3.5 onpick 事件

onpick 事件在定义时一般通过 onpick 属性指定一些项目，当用户选择或取消这些项目时，即可触发 onpick 事件，执行 onpick 属性所指定的项目，如打开卡片、卡片组或其他事件等。onpick 事件通常在 option 元素的标签中进行定义，其语法格式为：

```
<option value="value" onpick="href">
    内容(content)
</option>
```

可以看出，onpick 事件作为 option 元素的一个属性来定义具体的动作。这个属性即 onpick，它指定了事件触发时浏览器需要定位的超链(href)的 URL 地址。

下面我们给出一个使用 onpick 事件的例子。该程序运行后，首先在屏幕上显示出含有 Cat、Dog、Horse 三个项目的选择菜单。当用户选择其中某个菜单项后，浏览器就会打开 onpick 属性指定的卡片组，也就是激活了 onpick 事件。例如，用户选择了第 1 个菜单项，则用户浏览器就会导航至 cat.wml 卡片组，并显示出其中的内容。程序如下：

```
<card id="Card_1">
    <p>
        Select your favorite animal:
        <select name="animal">
            <option value="1" onpick="cat.wml"> Cat </option>
            <option value="2" onpick="dog.wml"> Dog </option>
            <option value="3" onpick="horse.wml"> Horse </option>
        </select>
    </p>
</card>
```

6.3.6 onevent 元素

onevent 元素通过<onevent>和<onevent/>标签可以把包含的任务(task)与特定的事件捆绑在一起。当用户激活这一特定事件时，onevent 元素所绑定的任务就会被立即执行。onevent



元素的语法格式如下：

```
<onevent type="type">
    任务(task)
</onevent>
```

其中 task 是与 onevent 元素关联的动作，也是条件激活时浏览器即将执行的内容。与 do 元素一样，onevent 元素中用户也必须绑定且只能绑定 go、prev、noop 或 refresh 四种元素所实现任务中的一个任务(task)。go 元素用于定位指定的 URL 地址，prev 元素用于定位并打开前一操作或任务，noop 为空操作，refresh 用于刷新当前卡片组或任务。

onevent 元素只有一个属性，即 type 属性，它是必选属性，主要用于定义特定事件的名称。属性值的数据类型为 CDATA 型。

下面的程序就是一个使用 onevent 元素的例子。当用户向回导航到当前卡片时，onenterbackward 事件就会被激活，这时浏览器中显示的并不是卡片中的“Welcome to China!”的信息，而是 onevent 元素为 onenterbackward 事件绑定的任务，即 www.chinawapnet.com 中 goodluck.wml 卡片的信息。程序清单如下：

```
<card>
    <onenterbackward type="onenterbackward">
        <go href="http://www.chinawapnet.com/goodluck.wml"/>
    </onenterbackward>
    <p>
        Welcome to China!
    </p>
</card>
```

下面我们给出一个结合使用 onevent 元素和 onpick 事件的例子。该程序可以让用户选择自己所在的城市，并把选择结果显示在另一卡片中：

```
<wml>
    <head>
        <meta http-equiv="Cache-Control" content="max-age=0"/>
    </head>
```

```
<card id="card0" ordered="false">
  <p>
    Please select a city...
    <select title="Cities List" name="city">
      <option title="Beijing" value="Beijing">
        <onevent type="onpick">
          <go href="#card01"/>
        </onevent>Beijing</option>
      <option title="Shanghai" value="Shanghai" onpick="#card01">Shanghai</option>
      <option title="Hongkong" value="Hongkong" onpick="#card01">Hongkong </option>
    </select>
  </p>
</card>

<card id="card01">
  <p>
    You are Living in $(city:noesc)
  </p>
</card>
</wml>
```

6.3.7 postfield 元素

postfield 元素用于指定当浏览器接到 URL 请求时，向源服务器(origin server)传送的域名及域值。传输时，传输域及传输值的实际编码方式主要依赖于浏览器与源服务器的通信方式。postfield 元素是通过单独的<postfield/>标签进行定义的，其语法格式如下：

```
<postfield name="name" value="value"/>
```

它共有两个属性：name 与 value，它们的取值均属于 VDATA 型数据。其中，name 属性用于指定传输域的名称，value 属性用于定义传输域的值。这两个属性均为必选属性。

下面我们给出一个使用 postfield 元素的例子。该程序用于向 Web 服务器发送 3 个域名及域值，并允许用户从浏览器端发回数据。这里发送数据的方式采用的是 HTTP POST 方式。程序如下：



```
<go method="post" href="http://hostname/servlet/bank">
  <postfield name="money" value="100"/>
  <postfield name="account" value="12345"/>
  <postfield name="operation" value="deposit"/>
</go>
```

6.3.8 卡片与卡片组的任务替代

前文述及，在 WML 程序中，我们可以使用多种元素为某一卡片绑定所需的事件，绑定时可以在卡片一级实现，也可以在卡片组一级实现。

在卡片一级绑定时，事件处理元素需包含在 card 元素中，并需指定该事件针对当前卡片的具体处理行为。在卡片组一级绑定时，事件处理元素需包含在 template 元素中，且需指定针对当前卡片组中所有卡片的具体处理行为。卡片组一级的事件处理元素在定义上相当于给其中所有卡片指定以相同的事件元素。

不过，在个别情况下，尽管卡片组中定义了统一的事件处理元素，但某些卡片可能需要使用其他事件处理元素或事件的其他处理参数，此时，就需要考虑卡片与卡片组的任务替代问题了。前面我们曾经简单地提到过几次，这里我们给出一般的替代规则：

- (1) 如果指定的是同名事件，那么卡片一级的事件处理元素将替代卡片组一级的事件处理元素。
- (2) 如果定义的 type 属性相同，那么卡片一级的 onevent 元素将替代卡片组一级的 onevent 元素。
- (3) 如果 do 元素所绑定事件的名称相同，那么卡片一级的 do 元素将替代卡片组一级的 do 元素。
- (4) 如果卡片一级的事件元素要替代卡片组一级的事件元素，而且前者的元素指定的是空(noop)任务，那么两者的事件元素都将被屏蔽起来，后者指定的任务也不会被浏览器执行和显示。

下面给出一个替代的例子，程序如下：

```
<wml>
  <template>
    <do type="options" name="do1" label="default">
```

```
<prev/>

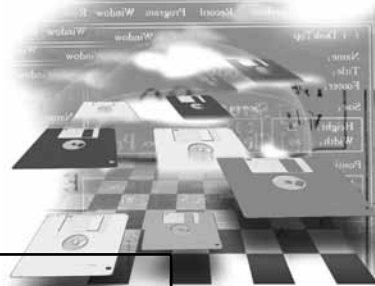
</do>
</template>

<card id="first">
  <!--该卡片将自动套用模板中定义的事件处理过程-->
  <!--该卡片的其余内容-->
  ...
</card>


<card id="second">
  <!--该卡片将使用空操作(noop)来替代模板中定义的事件处理过程，也就是说，该卡片中
    不再执行模板中的事件-->
  <do type="options" name="do1">
    <noop/>
  </do>
  <!--该卡片的其余内容-->
  ...
</card>

<card id="third">
  <!--该卡片使用同名的事件处理替代了模板中提供的事件处理-->
  <do type="options" name="do1" label="options">
    <go href="/options"/>
  </do>
  <!--该卡片的其余内容-->
  ...
</card>
</wml>
```

在这一程序中，卡片组一级定义了一个模板，它包含有 do 元素，指定当用户从选项(options)中选择事件“do1”时，执行 prev 任务，打开系统默认的卡片。该模板绑定的事件将同时应用于当前卡片组中的所有卡片中。随后，程序定义了卡片“first”，它自动套用了刚刚定义的模板。接下来，程序又定义了卡片“second”，它不想执行模板中的事件，为此该卡片使用空操作(noop)来替代模板中定义的事件处理过程，也就是说，该卡片中不再执行



模板中的事件。接下来，程序又定义了卡片“third”，并使用同名的事件处理替代了模板中定义的事件处理，即当用户选择“do1”时，该卡片将执行`<go href="/options"/>`而不是`<prev/>`。

需要指出的是，在卡片和卡片组中，我们也可以采用上述方法对 `onenterforward` 和 `onenterbackward` 元素所指定的内在事件进行替代；对于卡片组中使用 `onevent` 元素指定的事件处理过程，我们也可以使用 `onenterforward` 元素或 `onenterbackward` 元素对它进行替代，反之亦然。

下面，我们结合前几节的内容，给出一个任务替代和卡片组间导航的例子。这一例子中使用了 `access`、`head`、`meta`、`template`、`noop` 等元素，程序如下：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1/EN" "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <template>
    <do type="prev" name="Previous" label="Previous">
      <prev/>
    </do>
  </template>

  <card id="card1" title="First Card" newcontext="true">
    <p>
      Card 1 ... <br/>
      <do type="accept" label="Next Card">
        <go href="#card2"/>
      </do>
      <!--这里必须替代 do 元素中的 prev 任务，否则浏览器就会试图执行该任务。因为对
        第 1 个卡片来说，它的前面没有卡片，即 prev 任务不能执行，所以我们必须把
        它替代成其他任务。在这里，我们把它替代成 go 元素定位第 2 个卡片。-->
      <do type="prev" name="Previous">
        <noop/>
      </do>
    </p>
  </card>
```

```
<card id="card2" title="Second Card">
  <p>
    Card 2 ... <br/>
    <do type="accept" label="Next Card">
      <go href="#card3"/>
    </do>
  </p>
</card>

<card id="card3" title="Third Card">
  <p>
    Card 3 ... <br/>
    <!-- 注意，下面的 do 元素定位到 go 元素指定的 URL 地址，即另一 WML 程序
         deckno2.wml，它是另一个卡片组。 -->
    <do type="accept" label="Next Deck">
      <go href="deckno2.wml"/>
    </do>
  </p>
</card>
</wml>
```

该程序的执行情况说明如下：

(1) 首先进入当前卡片组的第 1 张卡片，由于模板中的 do 元素功能被替代，所以只显示出第 1 张卡片的提示信息及“Next Card”的选项，如图 6.3(a)。模板中的“Previous”选项没有显示，它被替代掉了。

(2) 选择“Next Card”选项后，当前卡片组的第 2 张卡片出现在手机屏幕上，如图 6.3(b)所示。可以看到，它含有“Next Card”和“Previous”两个选项。

(3) 从中再次选择“Next Card”选项后，当前卡片组的第 3 张卡片便出现在屏幕上，如图 6.3(c)所示。可以看到，它含有“Next Deck”和“Previous”两个选项。

(4) 此时如果选择“Next Deck”选项，则浏览器就会被定位到另一卡片组 deckno2.wml，并把它的信息显示在手机屏幕上，如图 6.3(d)所示。

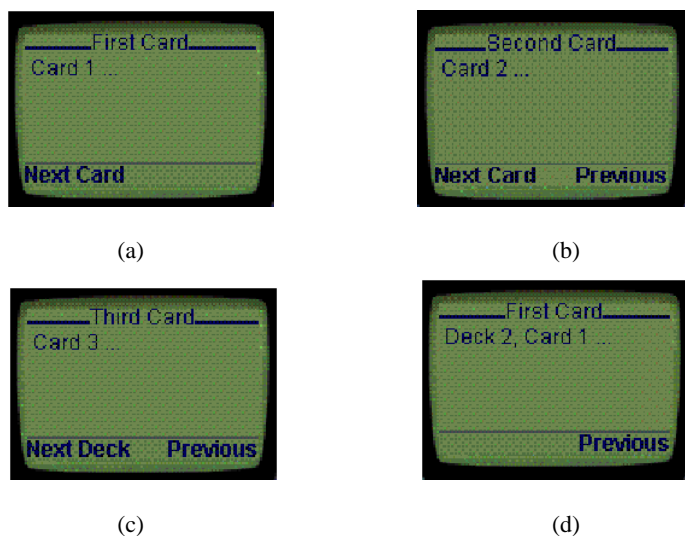


图 6.3 任务替代和卡片组间导航的实例

6.4 变量设置元素与变量设置的有关规定

几乎所有的 WML 内容都可通过设置参数来实现,这为我们灵活地开发 WML 程序提供了方便。例如,我们可以设置卡片的有关参数,从而可以很容易地改变卡片的显示样式或内容;我们也可以设置导航参数,以使浏览器能够根据用户的输入内容进行导航并显示相应的卡片内容。在 WML 编程中,参数设置都是通过变量设置和变量赋值等操作实现的。变量可以用来替换字符串,并可在程序运行中替代它们的当前值。只要变量的值不为空字符串,且变量在当前上下文内容中不是没有被定义或被认同,那么我们就可以使用一个非空的字符串去设置这个变量,并让它在程序中发挥作用。

本节我们先介绍一个变量设置元素,然后再介绍与变量设置有关的一些具体规定。

6.4.1 setvar 元素

setvar 元素用于指定在当前上下文内容中的变量的值,从侧面影响正在运行的任务。其语法格式为:

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

它有两个属性: name 和 value。前者用于指定变量的名称,后者用于指定所需赋给变量的值。这两个属性都是必选的,它们的数据类型均属于 VDATA 型。如果 name 属性所规定

的变量名不合法或不符合运行环境的要求,那么 setvar 元素在 WML 程序运行中就会被忽略,不能发挥其应有的作用。

现在我们看一个包含 setvar 元素的例子,程序如下:

```
<card id="Product" title="Choose Product">
  <p>
    Product name:
    <input title="Product name" name="product" value="WapSdk"/>
    <do type="accept" label="Clear">
      <refresh>
        <setvar name="product" value=""/>
      </refresh>
    </do>
  </p>
</card>
```

在这个程序中,我们使用 input 元素指定了名为 product 的变量名,并赋给它默认值 WapSdk,同时提供了一个“Clear”选项。当用户选择该选项时,setvar 元素定义的内容就会发生作用,由于它给变量 product 赋予空值,所以此时 product 原有的值就会被清除,同时使用空值来刷新当前卡片内容的显示。

6.4.2 变量设置

WML 编程中可以使用变量,变量使用前必须进行定义。变量的命名原则及定义方式我们上一章已经讲过(参见 5.3.3 节),这里不再重述。在这里,我们主要介绍 WML 程序中设置变量的一些规定。

如前所述, setvar 元素可用来设置变量,设置时 setvar 元素一般需要在 go、prev 或 refresh 元素中进行定义。另外,利用 input 和 select 元素也可以设置变量。其中前者是将用户输入的文本赋给变量,作为变量的值;而后者则将用户从 option 元素中选择的 value 属性的值赋给变量。有关 input 元素和 select 元素的语法格式及具体用法我们后面再行介绍。

设置变量时,以下几种情况还应当引起大家注意:

(1) 可以使用 WMLScript 的变量值设置 WML 的变量,反之亦然。也就是说,使用 WML



及 WMLScript 编写程序时，它们可以使用同名变量。

(2) 在 WAP 开发工具中，通常提供有对变量进行管理和维护的选项卡或对话框，开发人员从中也可以对相应的变量进行设置及编辑。

(3) 在当前上下文内容中，可以使用 card 元素的 newcontext 属性来清除所有的变量值。

6.4.3 变量定义和设置举例

为了便于大家深入了解 WML 编程中变量的定义和设置方法，我们这里给出一个稍微复杂的实例。该例中使用了 card、do、go、prev、setvar、onevent、refresh 等元素，并使用了 card 元素中的 newcontext 属性，定义了 3 个卡片，同时在这些卡片中定义了若干变量，给出了变量定义和设置的具体方法。程序如下：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM/DTD WML 1.1//EN" "http://www.wapforum.org/DTD/
wml_1.1.xml">

<wml>
  <card id="card1" title="First Card" newcontext="true">
    <p>
      Card 1 ... <br/>
      <!-- 在进入其他卡片前，下述变量将保持未定义状态-->
      card1 var1 = $(card1_var1) <br/>
      card2 var1 = $(card2_var1) <br/>
      card3 var1 = $(card3_var1) <br/>
      <do type="accept" label="Next Card">
        <go href="#card2">
          <setvar name="card1_var1" value="val_1"/>
        </go>
      </do>
    </p>
  </card>

  <card id="card2" title="Second Card">
    <p>
      Card 2 ... <br/>
      card1 var1 = $(card1_var1) <br/>
```



```
card2 var1 = $(card2_var1) <br/>
card3 var1 = $(card3_var1) <br/>
<do type="accept" label="First Card">
    <go href="#card1"/>
</do>

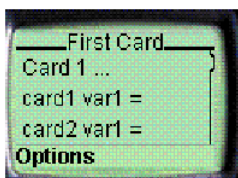
<do type="accept" label="Third Card">
    <go href="#card3">
        <setvar name="card2_var1" value="val_2"/>
    </go>
</do>

<do type="prev" label="Previous Card">
    <prev/>
</do>
</p>
</card>

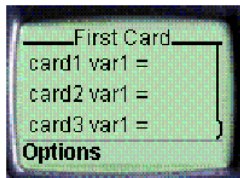
<card id="card3" title="Third Card">
    <onevent type="onenterforward">
        <refresh>
            <setvar name="card3_var1" value="val_3"/>
        </refresh>
    </onevent>
<p>
Card 3 ... <br/>
card1 var1 = $(card1_var1) <br/>
card2 var1 = $(card2_var1) <br/>
card3 var1 = $(card3_var1) <br/>
<do type="prev" label="Previous">
    <prev/>
</do>
</p>
</card>
</wml>
```

该程序执行结果说明如下：

(1) 程序在 WAP 手机的浏览器中运行后, 首先显示第 1 张卡片的内容, 包含卡片标题及当前卡片中定义的 3 个变量, 此时变量并没有赋值, 如图 6.4 所示。由于手机屏幕较小, 用户可使用手机操作面板上的向下滚动按键, 浏览当前卡片的其他内容。



(a) 第 1 屏



(b) 第 2 屏

图 6.4 第 1 张卡片内容

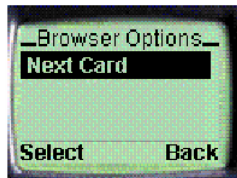
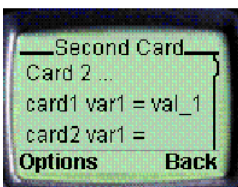


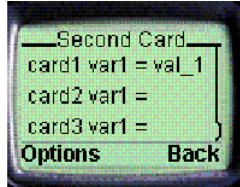
图 6.5 选择 “Next Card”

(2) 接下来, 在手机上按一下 “选项(Options)” 按钮, 手机屏幕上出现图 6.5 所示的选项, 从中将选择光标移向 “下一卡片(Next Card)” 一项。

(3) 然后按一下手机上的 “选择(Select)” 按钮, 选择 “下一卡片(Next Card)” 项, 程序即进入第 2 张卡片, 手机浏览器即显示出第 2 张卡片的内容。此时, 第 1 张卡片中的 setvar 元素被执行, 变量 “card1 var1” 被赋予了 “val_1” 的变量值, 而其他两个变量仍未被赋值, 如图 6.6 所示。



(a) 第 1 屏



(b) 第 2 屏

图 6.6 第 2 张卡片内容

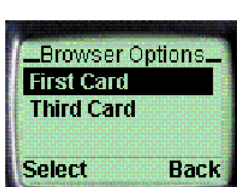
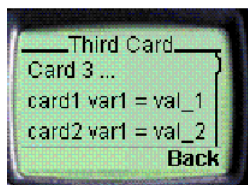


图 6.7 选择 “Third Card”

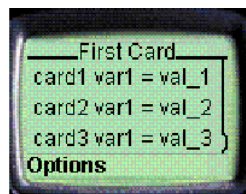
(4) 再在手机上按一下 “选项(Options)” 按钮, 手机屏幕上就会出现第 2 张卡片中由 do 元素指定的选项, 如图 6.7 所示。从中将选择光标移向 “First Card” 或 “Third Card” 选项, 并按手机上的 “选择(Select)” 按钮, 即可分别进入第 1 张卡片和第 3 张卡片, 浏览器中就会显示出它们各自的内容。

(5) 我们这里选择 “Third Card” 选项, 则第 2 张卡片中的 setvar 元素被执行, 变量 “card1 var2” 被赋予了 “val_2” 的变量值, 而变量 “card1 var3” 尚未被赋值。随后, 浏览器进入并显示第 3 张卡片。此时, 第 3 张卡片中的 setvar 元素被执行, 变量 “card1 var3” 被赋予了 “val_3” 的变量值。因此, 第 3 张卡片显示时, 我们可以浏览到 3 个变量均已被赋值的情况, 如图 6.8(a)所示。如果从中选择两次 “Back”, 则可进入第 1 张卡片。可以看到, 现

在第 1 张卡片中, 3 个变量均已有值, 如图 6.8(b)所示。



(a) 第 3 张卡片



(b) 第 1 张卡片

图 6.8 3 个变量均已有值的情形

6.5 用户输入处理元素

通过 WAP 手机的按键, 用户可以向浏览器显示的卡片中输入数据信息或操作信息, WML 为此专门提供了处理用户输入的元素, 如 input、select、option、optgroup 和 fieldset 等, 本节就详细讨论这些元素的功能与用法, 以及 WML 程序中处理用户输入的编程方法。

6.5.1 input 元素

input 元素用于定义文本实体对象, 包含有对输入文本内容的格式、数据类型、长度、值、变量名等多种属性的具体规定。当用户输入满足 input 元素的规定时, 则接受输入信息, 并赋给指定的变量或进行相应的操作、处理; 否则, 就通过浏览器给出具体的处理意见, 并进行适当的输入处理或变量初始化操作, 比如刷新卡片以让用户重新输入, 或给用户指出输入错误所在并等待进一步的处理指令等。input 元素是 WML 编程中处理用户交互活动的重要元素, 它通过单独的<input/>标签进行定义, 语法格式如下:

```
<input name="variable" title="label" type="type" value="value" default="default"
      format="specifier" emptyok="false|true" size="n" maxlength="n" tabindex="n"/>
```

其中除了 name 属性是必选的以外, 其他属性都是可选的。这些属性的功能和用法介绍如下:

(1) name。该属性用于指定用来保存用户输入文本的变量的名称。定义 name 属性后 WML 将根据该属性的值也即变量名, 为即将输入的文本实体对象预置存储空间, 以便接收用户输入。

(2) title。该属性用于定义 input 元素的标题，通常是位于输入框前的提示信息。

(3) type。用于指定文本输入区的类型，有 text 和 password 两种选择。默认值为 text，指定用户可以输入文本，而且输入的文本会同时逐键响应并显示在浏览器中。如果选择 password，则指定用户输入的文本作为密码文本处理，WML 程序按文本实体接受输入的数据，而浏览器上响应用户输入显示时逐键均为星号(*)，由此起到保密的目的。

例如，当 type="text"时，用户输入“12345”，那么浏览器上将依次显示出“12345”；如果 type="password"，则对用户输入的上述内容，浏览器依次显示为“*****”。这两种情况下，WML 程序接收到的用户输入都是“12345”，并把它赋给相应的变量。

(4) value。该属性用于指定 name 属性所定义变量的值，它将显示在输入框中。如果 input 元素显示时用户没有设置 name 属性中的变量，那么 value 属性中的值就会自动分配给该变量。如果 name 属性中的变量已有值，或用户输入了新的值，或者用户输入的值不符合 input 元素的格式要求，那么 value 属性就会被程序忽略，value 属性定义的值也不会影响变量的值。value 属性在语法和行为上与下面介绍的 default 属性基本相同。

(5) default。该属性用于指定 name 属性所定义变量的默认值。在 input 元素运行时，如果用户不输入变量值，那么 name 属性定义的变量将采用 default 属性定义的默认值。否则，用户输入新值时，默认值将被忽略，变量将采用新输入的值。如果用户输入的值不符合 input 元素的格式要求，那么变量仍采用默认值。

表6.1 格式化标记及作用

标记	描 述
A	允许输入任何大写字母及标点符号，即非数字字符的大写字母和符号
a	允许输入任何小写字母及标点符号，即非数字字符的小写字母和符号
N	允许输入任何数字，不包括特殊符号及大小写字母
X	允许输入任何符号、数字及大写字母，所输入字母不可改变为小写
x	允许输入任何符号、数字及小写字母，其中所输入字母不可改变为大写
M	允许输入任何符号、数字及大写字母，所输入字母可改变为小写，默认为首字大写
m	允许输入任何符号、数字及小写字母，所输入字母可改变为大写，默认为首字小写
*f	允许输入任何格式的任意个字符(但不超过 maxlength 属性所定义的个数)，其中 f 为上述任一格式标记代码，比如*N 可代表任何数字；*f 在格式串中只可指定一次且只能出现在最后
nf	允许输入 n 个字符，其中 n 是 1~9 的整数；f 为除*f 以外的上述任一格式标记代码；nf 在格式串中也是只可指定一次且只能出现在最后
\c	允许输入一个字符，并显示在输入框光标所在的当前位置

(6) format。该属性用于格式化输入的数据。这种格式化是通过格式标记实现的，可用

的标记有两种形式，一是“一位数字标记”形式，二是“*标记”形式。其中前者代表 N 个标记型字符，如 3X，而后者代表最多不超过 maxlength 属性值所对应个数的标记型字符。format 属性中可用的格式化标记及其作用描述，列在表 6.1 中，供大家参考。

(7) maxlength。该属性用于指定用户可输入字符串的最大长度，也即可输入字符的最大个数。该属性取值的上限为 256，即最多不得超过 256 个字符。

(8) emptyok。用于指定用户是否可以不在输入框内输入内容。它有 true 和 false 两种选择，默认为 false。emptyok 取值 false 时，表明用户需要输入内容，且输入的内容要符合格式、长度等属性要求。emptyok 取值为 true 时，表明用户可以不输入内容，此时，input 元素将接受一个空串作为输入内容。

(9) size。该属性用于指定输入框的宽度，宽度值为字符个数。

(10) tabindex。用于指定多个输入框存在时，类似于 HTML 中 Tab 键的具体位置。即通过该键所在的输入框位置，间接定义当前输入操作发生在哪一个输入框。这样，用户可以通过 tabindex 的值，确定欲输入内容的项目。当然，待输入各项在此之前需要编号，以便于 input 元素根据 tabindex 的值进行定位，编号越大，越排在输入顺序的后面。

下面我们通过几个实例具体分析一下 input 元素的用法。例如：<input name="X" type="text" maxlength="32"/>一句定义了变量 X，指定 input 元素可以接受任意字符，并同时显示出用户输入的文本字符，可输入文本字符串的最大长度为 32 个字符，输入的文本将赋给 X 变量。

再如，<input name="name" type="text" value="Robert"/>定义了一个名为 name 的变量，并指定了初始值“Robert”，输入类型为文本，输入时会同时显示出来。

再看下面一个例子。它定义了一个输入用户的名(First name)、姓(Last name)及年龄(Age)的卡片，其中年龄(Age)输入时需要输入两个数字，因为它的输入格式为“NN”。

```
<card>
  <p>
    First name: <input type="text" name="first"/><br/>
    Last name: <input type="text" name="last"/><br/>
    Age: <input type="text" name="age" format="NN"/>
  </p>
</card>
```



下面我们再给出一个含有输入口令密码的例子，程序比较简单，我们不再分析介绍。

```
<wml>
  <card>
    <p>
      First Name: <input name="fname" maxlength="15" />
      Last Name:  <input name="lname" maxlength="15" />
      State:      <input name="state" maxlength="2" emptyok="true" value="CA" />
      Zipcode:    <input name="zipcode" maxlength="9" />
      Password:   <input name="password" maxlength="8" type="password" />
    </p>
  </card>
</wml>
```

6.5.2 select 元素

选择列表属于输入元素，允许用户从选项列表中选择需要的项目。WML 不仅支持单选列表，即单选项，而且支持多选列表，也就是复选项。select 元素允许用户从选项列表中选择所需的项目。列表中的选项采用后面我们就要讲到的 option 元素进行定义，一般是一行格式化的文本。编程时，我们可以使用 optgroup 元素将 option 元素的项目分成不同级别或层次的选项组，以求为用户选择提供方便。有关 optgroup 元素的情况我们随后介绍。

select 元素是通过<select>和</select>标签进行定义的，语法格式如下：

```
<select title="label" multiple="false|true" name="variable" default="default" iname="index_var"
  ivalue="default" tabindex="n" >
  内容(content)
</select>
```

其中所有属性都是可选的。实际编程中，<select>和</select>标签之间的内容中通常还包含有 optgroup 元素和 option 元素，这时的语法格式为：

```
<select title="label" multiple="false|true" name="variable" default="default" iname="index_var"
  ivalue="default" tabindex="n" >
  <optgroup title="label"> 选单内容(content) </optgroup>
```

```
<option title="label" value="value" onpick="url"> 内容(content) </option>
其他内容(content)
</select>
```

select 元素各个属性的功能和用法介绍如下：

(1) multiple。该属性用于指定选择列表是否可以使用复选项，有 false 和 true 两种选择，默认值为 false。当设置为 true 时，用户可以从选择列表中选择多个选项，即选择列表中使用复选项。当设置为 false 时，用户只可从列表选择一个选项，即单选项。

(2) name。该属性用于指定接收选项值的变量的名称，变量值由 value 属性预设。

(3) value。用于指定 name 属性所定义变量的默认值。如果 select 元素显示时，name 属性定义的变量没有值，用户浏览器则按照此默认值显示；如果 name 变量已经有值，则忽略 value 的值。在用户做出选择之前，即在选择选项之前，由于 name 变量还没有得到与选项对应的值，所以这时变量值将采用 value 默认值。

如果 select 元素允许复选项的多项选择，那么用户的选择结果将是多个选项组成的字符串，相邻选项之间使用分号(;)间隔，如“cat;mouse;dog;horse”，这个字符串将作为 name 变量的值。这种情况下 value 为变量预置的默认值，也将是一个使用分号(;)间隔的字符串，即格式要对应。另外，选项组成的字符串之间不能有空格，如“cat;mouse; ;dog;horse”是不合法的，WML 一律按无效值处理。

(4) iname。用于指定包含排序号的变量的名称，也就是用于接受选择列表中选项的变量的名称。变量的值由 ivalue 属性指定。用户浏览器根据其中的排序号设置默认的选项，或者说确定选项的位置，比如，0 意味着选择列表中没有选项，1 指选择列表中的第 1 个选项，2 指第 2 个选项，其他数字和选项依次类推，并逐项递增，直到穷尽所有选项。

(5) ivalue。用于指定选择列表中被选中选项的值，是一个具有排序号性质的值。如果 iname 属性指定的变量在卡片显示时没有值，那么它们就采用 ivalue 指定的值，否则就忽略 ivalue 属性的值。

如果 select 元素允许复选项的多项选择，那么用户所选择项目的值将是多个选项值组成的数字串，相邻数字之间使用分号(;)间隔，如“1;2;3;4”，这个数字串将作为 iname 变量的值。这种情况下 ivalue 为变量预置的默认值，也将是一个使用分号(;)间隔的数字串，它们的格式也要对应。而且，数字串中不能有空格，如“1;2; ;3;4”是不合法的，WML 将对此按

无效变量值处理。

(6) title。用于指定选择列表的标题。

(7) tabindex。用于指定当前选择光标在选择列表中的具体位置,该位置即为当前选择操作将要选择的选项所在的位置。这样,用户可以通过 tabindex 的值,确定欲选择选项。当然,待输入各项在此之前需要编号,以便于 select 元素根据 tabindex 的值进行定位,编号越大,越排在被选顺序的后面。

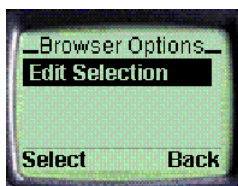
下面我们给出一个复选项选择列表的例子,程序如下:

```
<wml>
  <card>
    <p>
      Please choose <i>all</i> of your favorite animals:
      <select name="X" iname="I" ivalue="1;2" multiple="true">
        <option value="D"> Dog      </option>
        <option value="C"> Cat      </option>
        <option value="H"> Horse    </option>
      </select>
    </p>
  </card>
</wml>
```

其中 select 元素定义了用于获取选项的变量“X”,用于获取选项序号的变量“I”,指定了已选复选项变量值“1;2”,并指定 select 元素允许接受复选项。其中,已选复选项变量值“1;2”用于指定选择列表显示时,第1项和第2项选项为已选中状态。



(a)



(b)



(c)

图 6.9 复选项选择列表举例

在该程序中,变量 I 已经设置了初值,如果变量 I 没有进行预设置,那么用户就要预先设置“dog”和“cat”选项,使这两个复选项处于已选状态,设置方法是输入“D;C”,如图 6.9(a)所示。当然,如果用户选择“cat”和“horse”选项,则变量“X”就会被设置为“C;H”,

且变量“ I ”被设置为“ 1;3 ”。图 6.9(b)和(c)给出该程序另外两个操作界面。

6.5.3 option 元素

option 元素用于定义 select 元素中的一组单选项。它通过<option>和</option>标签进行定义，并可包括事件和单选项的显示文本等信息，其语法格式如下：

```
<option title="label" value="value" onpick="href" >  
    内容(content)  
</option>
```

option 元素的属性均为可选，各属性功能及用法说明如下：

(1) value。该属性用于设置键值。当用户选到该项后，option 元素就会将该键值赋给 select 元素的 name 属性所指定的变量。

(2) title。用于为 option 元素指定一个标题，以提示用户操作。

(3) onpick。该属性用于指定用户选到该项并按 ACCEPT 键后所打开卡片组的 URL。

下面这个例子使用 option 元素定义了一个简单的单选项列表。从中选择需要的选项，select 元素定义的变量就会获得 option 元素指定的变量值。比如，如果用户选择“ Dog ”选项，变量 X 就会获取值“ D ”。程序如下：

```
<wml>  
    <card>  
        <p>  
            Please choose your favorite animal:  
            <select name="X">  
                <option value="D"> Dog </option>  
                <option value="C"> Cat </option>  
            </select>  
        </p>  
    </card>  
</wml>
```

该程序的运行操作示意图如图 6.10 所示。

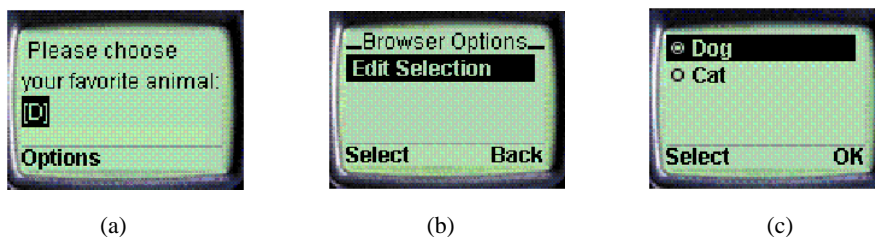


图 6.10 单选项选择列表举例

对于这种单选项列表，我们也可以设置某些选项的初始值。比如，在下面的程序中，我们想在单选项列表中预先选中“Shanghai”一项，那么就使用 select 元素定义变量 I，并给它赋初值“1”。该程序如下：

```
<wml>
  <card>
    <p>
      Please choose your favorite city:
      <select iname="I" ivalue="1">
        <option value="S"> Shanghai </option>
        <option value="B"> Beijing </option>
        <option value="H"> Hongkong </option>
      </select>
    </p>
  </card>
</wml>
```

6.5.4 optgroup 元素

optgroup 元素用于将多个相关的 option 元素进行分组，用户浏览器可以借助这种分组来安排选项列表的显示布局，以方便用户选择。optgroup 元素是通过<optgroup>和</optgroup>标签进行定义的，其语法格式如下：

```
<optgroup title="label" >
  内容(content)
</optgroup>
```

它所包含的内容(content)中需要包含至少一次 option 元素或其他 optgroup 元素。

optgroup 元素只有一个属性，即 title 属性，用于定义 optgroup 元素的标题，以方便用户选择和操作。

我们来看一个使用 optgroup 元素进行选项分组的例子。程序如下：

```
<wml>
  <card id="card1" title="Country">
    <p>
      Select a country:
      <select name="country" multiple="true" tabindex="2">
        <optgroup title="Scandinavia">
          <option value="den">Denmark</option>
          <option value="fin">Finland</option>
          <option value="nor">Norway </option>
          <option value="swe">Sweden </option>
        </optgroup>
        <optgroup title="Europe">
          <option value="fra">France </option>
          <option value="ger">Germany</option>
          <option value="ita">Italy </option>
          <option value="spa">Spain </option>
        </optgroup>
      </select>
    </p>
  </card>
</wml>
```

该程序中定义的卡片中包含有一个 select 元素实现的某地区国家的选择列表，首先是 optgroup 元素定义的两个选项组，即“斯堪第纳维亚(Scandinavia)”和“欧洲(Europe)”，它们分别又包含由 option 元素定义的两个下一级的选择列表，从中可以选择“Scandinavia”和“Europe”两个地区内的国家。Scandinavia 是指包括挪威、瑞典、丹麦、冰岛在内的北欧一地区。

程序运行时，将出现 optgroup 元素定义的两个选项组名称，如图 6.11(a)所示，从中选择一组，比如“Scandinavia”，则可列出这一地区的国家列表，如图 6.11(b)所示，从中可以选择所需的国家名。

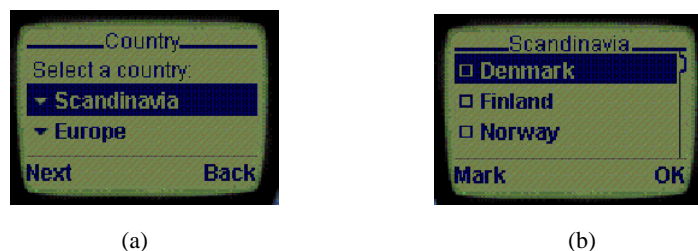


图 6.11 optgroup 元素使用举例

6.5.5 fieldset 元素

fieldset 元素用于设置输入框和相应的说明文本,从而用户就可以利用 input 元素等借助该输入框输入所需的数据信息。fieldset 元素的语法格式为:

```
<fieldset title="label" >
    内容(content)
</fieldset>
```

由于 fieldset 元素与输入有关,所以它的内容(content)中可以包含与输入有关的其他元素,如 input、select 等;而且还可以设置输入框提示信息文本的格式,如 em、strong、b、i、u、big、small 等,有关这些文本格式设置元素,我们后面会详细介绍的。

由其语法格式可以看出,fieldset 元素只有一个属性,即 title 属性,用于定义 fieldset 元素的标题,以方便用户选择和操作。

下面我们来分析一个使用 fieldset 元素的实例,程序如下:

```
<wml>
  <card id="info" title="Personal Info">
    <do type="accept" label="Submit">
      <go href="/submit?f=$(fname)&l=$(lname)&s=$(sex)&a=$(age)"/>
    </do>

    <p>
      <fieldset title="Name">
        First name: <input type="text" name="fname" maxlength="32"/><br/>
        Last name: <input type="text" name="lname" maxlength="32"/><br/>
      </fieldset>
    </p>
  </card>
</wml>
```

```
<fieldset title="Info">
    <select name="sex">
        <option value="F">Female</option>
        <option value="M">Male</option>
    </select>
    <br/>
    Age: <input type="text" name="age" format="*N"/>
</fieldset>
</p>
</card>
</wml>
```

这一程序定义了一个 WML 的卡片组，让用户输入简单的个人信息。这些个人信息被安排到 fieldset 元素定义的几个输入框中进行输入，输入框还有提示信息，以提示用户的输入操作。

6.6 锚、图像、定时器及其元素

本节我们讲解与定位或定时控制有关的 3 类元素，包括 anchor、a、img、timer 几种元素。使用它们可以在 WML 卡片中创建超链接，或在文本流中显示一幅图像，或设置定时器来控制用户操作及卡片显示等。

6.6.1 anchor 元素

anchor 元素用于创建一个超链接的头部，超链接的其余部分为用户指定的 URL 地址。当程序运行中用户选中该超链接时，浏览器即会被引入到超链接指定的地址，如其他卡片组或同一卡片组中的其他卡片。

anchor 元素由<anchor>和</anchor>标签进行定义，它所包含的超链接必须是真实存在的，而且是能够正确链接的超链接。anchor 元素定位超链接时，必须通过相关的任务元素完成定位处理，如 go 元素、prev 元素、refresh 元素等。不过，在 anchor 元素中只能包含 1 个定位任务，多于一个时会导致 WML 运行错误。

anchor 元素的语法格式如下：

```

<anchor title="label" >
    任务(task)
    文本(text)
</anchor>

```

其中的任务(task)需要包含一个进行定位的任务元素。可以看到，anchor 元素只有一个属性，即 title 属性，它用于定义 anchor 元素的超链接标题。用户浏览时可利用这一标题，来及时了解欲操作的超链接的名称或有关提示信息。为了适用于大多数的 WAP 手机浏览器，一般来说，title 属性指定的标题不应多于 6 个字符。

下面就是一个使用 anchor 元素的例子：

```

<wml>
  <card id="links" title="Links">
    <p>
      This is normal text, but here is a
      <anchor title="LINK">link!
        <go href="dir/file.wml">
          <setvar name="var_name" value="var_value"/>
        </go>
      </anchor>
    </p>
  </card>
</wml>

```

其中 anchor 元素定义了锚的标题“LINK”，它显示在浏览器的左下角，如图 6.12 所示；同时还定义了超链接的文本“link!”（见图 6.12），用户选择该超链接，浏览器即可被定位到由 go 元素指定的超链接任务，显示“dir”目录下的“file.wml”。

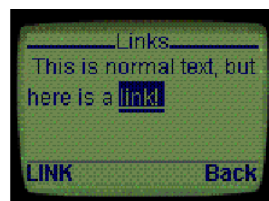


图 6.12 anchor 元素使用举例

6.6.2 a 元素

a 元素是 anchor 元素的简化形式，它内嵌了 anchor 元素需要包含的 go 元素功能来完成超链接定位，并且不再包含其他任何变量设置。它使用<a>和标签进行定义。例如，原

本利用 anchor 元素实现超链接的几行程序：

```
<anchor>follow me  
  <go href="destination.wml"/>  
</anchor>
```

可以使用 a 元素来实现，程序十分简单：

```
<a href="destination.wml"> follow me</a>
```

其中定义超链接的 href="destination.wml"也可以写在<a>与标签包含的内容中。因此，a 元素的语法格式有两种：

```
<a  title="label" >  
  href="href"  
  文本(text)  
</a>
```

或：

```
<a  href="href"> 文本(text) </a>
```

第 1 种格式中有一个 title 属性，它用于定义 a 元素的超链接标题，与 anchor 元素的 title 属性完全相同。为适应大多数的 WAP 手机浏览器，title 属性所指定的标题也不应多于 6 个字符。

6.6.3 img 元素

img 元素用于在格式化的文本中放置和显示一幅图像。当然，前提是用户所用浏览器必须支持图像显示。img 元素由单独的标签进行定义，它不包含其他元素。其语法格式如下：

```

```

属性中 alt 和 src 是必须要有的，其他可选。另外，需要注意的是 img 元素要放在 p 元



素里，而不能放在 do 或 option 等元素里。

img 元素各个属性的功能和用法介绍如下：

(1) alt。该属性用来指定当手机不支持图像显示时用来替换显示的文本文字。

(2) src。该属性用于指定图像文件的 URL 地址。当程序运行时，用户浏览器就会根据这一 URL 地址下载并显示图像文件。不过，当指定了下述 localsrc 属性时，手机浏览器则会忽略 src 属性。

(3) localsrc。该属性用来指定显示存在手机 ROM 中的图标文件。如果在 ROM 中找不到，则到服务器上去寻找。如果找得到，则显示该图标；否则，就显示 src 属性指定的图像。

(4) align。该属性用来指定图像显示时相对当前文本行的对齐方式，有 3 种选择：top、middle 或 bottom，默认值为 bottom。top 是指显示时图像顶边与当前文本的上边缘线水平对齐；middle 是指显示时图像中心与当前文本的中心线水平对齐；bottom 是指显示时图像底边与当前文本行的基线水平对齐。

(5) height。用于设定图像显示时的高度。如果图像的实际高度与 height 属性定义的高度并不相等，那么浏览器显示时将对图像进行适当的放大或缩小，使之满足该属性要求。height 属性设置时也可以是百分比数，此时它指定图像显示时的缩放百分比。

(6) width。与 height 属性类似，用于设定图像显示时的宽度或宽度百分比。显示时，浏览器将据此适当地放大或缩小图像，使之满足该属性要求。

(7) vspace。该属性用于指定图像显示时的上边距和下边距，默认值为 0。其数值不得超过卡片的高度。vspace 属性也可以取百分比，即占卡片高度的百分比，浏览器显示时将根据该百分比决定上、下边距的大小。

(8) hspace。与 vspace 属性类似，该属性用于指定图像显示时的左边距和右边距，可以是数值，也可以是百分比，默认值为 0。其数值不得超过卡片的宽度。

下面的一句程序给出了使用 img 元素的简单例子：

```

```

其中，src="bitmaps/moon.wbmp"指定显示 bitmaps 目录下的 moon.wbmp 图像文件，显示时左、右、上、下边距均为 1，即 1 个空格；如果用户手机的浏览器不支持图像显示，则显示文字“Moon”。

6.6.4 timer 元素

timer 元素用于设定一个定时器，可以延时显示卡片组、卡片，或实现 WML 程序的等待操作，或在卡片组和卡片之间实现切换以取得动画效果。

一个卡片只能使用一次 timer 元素，也就是说只能设置一个定时器。当用户进入含有定时器的卡片时，定时器即会开始工作，其时间值就会逐渐减少。timer 元素指定的时间值单位为 1/10 秒，时间值必须为正整数，当时间值减少到 0 时，定时器功能就会结束，此时，WML 将激活由 ontimer 元素指定的事件。timer 元素是由单独的<timer/>标签进行定义的，其语法格式如下：

```
<timer name="variable" value="value" />
```

它的两个属性中，value 属性是必选的，name 属性为可选。name 属性用于指定表示时间值的变量的名称，该变量的取值由定时器的时间值决定，时间值减少，该变量的值也相应的减少，并始终保持相等。当定时器时间减为 0 时，该变量的值也变为 0。

value 属性用于指定 name 属性所定义变量的初始值。如果 name 属性定义的变量在定时器初始化时还没有值，那么该变量就将采用 value 属性指定的值，否则，该变量就会忽略 value 属性的值。如果没有定义 name 属性，也就是说，没有指定时间变量，那么 timer 元素指定的定时器仍将采用 value 属性的值进行延时处理。

下面的程序就是使用 timer 元素的例子。当前卡片在显示“Hello World!” 10 秒钟后，浏览器将激活 ontimer 事件，自动引导至“/next”所指定的位置：

```
<wml>
  <card ontimer="/next">
    <timer value="100"/>
    <p>
      Hello World !
    </p>
  </card>
</wml>
```

其中的 ontimer 事件也可使用 onevent 元素和 go 元素结合实现，具体的程序如下：



```
<wml>
  <card>
    <onevent type="ontimer">
      <go href="/next"/>
    </onevent>
    <timer value="100"/>
    <p>
      Hello World!
    </p>
  </card>
</wml>
```

下面我们再看一个由变量实现定时的例子。程序中在 timer 元素内定义了变量 t，并指定了定时值 50，也就是 5 秒。每次进入这一卡片时，timer 都由变量 t 的值设定定时值，如果变量 t 没有设置具体的值，那么 timer 定时值将采用指定的定时值，即 value 属性指定的 5 秒。程序如下：

```
<wml>
  <card ontimer="/next">
    <timer name="t" value="50"/>
    <p>
      Hello World!
    </p>
  </card>
</wml>
```

最后我们再来看一个例子。该程序通过 timer 元素控制自动显示两个卡片的内容，两卡片显示时中间等待 5 秒钟。注意它使用 onevent 元素和 ontimer 事件控制从第 2 个卡片到第 1 个卡片的显示：

```
<wml>
  <head>
    <meta http-equiv="Cache-Control" content="max-age=0"/>
  </head>
```

```
<card id="card1" ontimer="#card2">
  <timer name="time1" value="50"/>
  <p align="center">
    After 5s, goto card2
  </p>
</card>

<card id="card2">
  <onevent type="ontimer">
    <go href="#card1"/>
  </onevent>
  <timer name="time2" value="50"/>
  <p align="center">
    Here is card2!
  </p>
</card>
</wml>
```

6.7 文本格式化及其元素

WML 程序中，为使显示的文本呈现出丰富的样式，WML 提供了一些用于格式化的元素，我们通过这些元素及其相应的标签可以对文本进行标注和控制，从而实现不同的显示效果。WML 控制文本时基本上采用了 XML 的一般原则，比如，遇有显示文本中含有连续的多个空格时，WML 的浏览器就会把这些连续的空格显示为单词间的一个空格；WML 还将空格、制表符和换行都显示为空白，等等。本节我们就讲解 WML 格式化元素的功能及具体用法。

6.7.1 增强元素

增强元素都是一些成对的标签，用于指定文本的增强显示信息。比如，b 元素通过和标签可以控制其中的文本按照粗体字进行显示。表 6.2 给出了 WML 增强元素的名称、标签及功能解释。

下面的程序中我们使用了几种增强元素，其显示效果如图 6.13 所示。


```

<wml>
  <card id="card1">
    <p>
      <em>
        A
        <u>
          Demonstration
        </u>
        of Nokia's
        <i>
          <strong> Wireless Application Protocol<br/> </strong>
        </i>
        <b> Toolkit</b>
      </em>
    </p>
  </card>
</wml>

```

表6.2 WML的增强元素

元素	标签	语法格式	功能
em	和	 文本(text) 	指定增强显示文本
strong	和	 文本(text) 	进一步加强文本的增强显示
i	<i>和</i>	<i> 文本(text) </i>	使用斜体字显示文本
b	和	 文本(text) 	使用粗体字显示文本
u	<u>和</u>	<u> 文本(text) </u>	显示文本时增加下划线
big	<big>和</big>	<big> 文本(text) </big>	使用大字体显示文本
small	<small>和</small>	<small> 文本(text) </small>	使用小字体显示文本

 为便于大家记忆，我们给出表 6.2 中几种增强元素的原单词解释：em 即 emphasis(增强、强调)，b 即 bold(粗体)，i 即 italic(斜体)，u 即 underlined(下划线)。

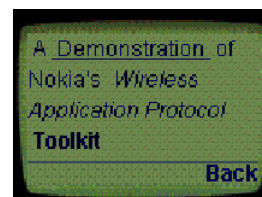


图 6.13 增强元素使用举例

6.7.2 br 元素

“br”即 break，是用于换行的元素，它是使用单独的
标签进行定义的。br 元素的

作用相当于插入一个回车符，并开始一个新行。其语法格式为：

```
<br/>
```

例如，下面的程序中就使用 br 元素插入了几个新行：

```
<wml>
  <card id="card1">
    WML is no real <br/>language such as <br/>C++ or Pascal, <br/>
    it is just a system <br/>for describing <br/>documents. <br/>
  </card>
</wml>
```

6.7.3 p 元素

“p”即指 paragraph(段落)，p 元素用于划分段落，使当前文本换行并插入一个空白行。p 元素可以使用单独的<p/>标签进行定义，也可以使用<p>和</p>标签成对地进行定义。其语法格式为：

```
<p align="alignment" mode="wrapmode" />
```

或

```
<p align="alignment" mode="wrapmode" >
  文本(text)
</p>
```

p 元素有两个属性，功能和用法介绍如下：

(1) align。该属性用于设置段落在浏览器中的对齐方式，有 left、center 和 right 三种取值。这三种参数值分别表示 p 元素当前定义的文本段落与浏览器窗口的左侧、中间和右侧进行对齐。默认值为 left，即段落与浏览器窗口的左侧对齐排列。

(2) mode。该属性用于指定下一段落的换行方式，有 wrap 和 nowrap 两种选择。wrap 指定下段文本换行，nowrap 指定下段文本不换行。如果不指定 mode 属性，则下一段的换行方式将采用与上一段相同的方式。卡片中默认的 mode 属性值为 wrap。

作为举例，我们给出一个使用 p 元素实现段落居中对齐、左对齐、文本可换行及文本不可换行的例子，程序如下：



```
<p align="center">
    centered text
</p>
```

```
<p align="left">
    left-justified
</p>
```

```
<p mode="wrap">
    This text is wrapped to the next line.
</p>
```

```
<p mode="nowrap">
    This text is truncated.
</p>
```

6.7.4 td 元素

td 元素用于规定表格单元格的内容。它使用成对的<td>和</td>标签进行定义，这两个标签之间的内容就是该单元格中的具体数据。当表格中规定了行之后，td 元素实现的实际是定义表格的列的功能。td 元素的语法格式为：

```
<td> 单元格内容(content) </td>
```

比如：<td> Hello! </td>一句程序指定了表格当前单元格中的内容为“Hello!”。有关 td 元素的进一步应用可参考后面有关 tr 元素和 table 元素的介绍。

6.7.5 tr 元素

WML 中的表格是按照行、列进行组织的。一个表格由若干行组成，每行由若干列组成。tr 元素用于定义表格的行，它使用成对的<tr>和</tr>标签将其中包含的内容定义为表格的一行。由于表格有行也有列，所以 tr 元素通常与 td 元素结合使用，td 元素用于定义表格的列。

tr 元素的语法格式如下：

```
<tr>
    <td> 单元格内容(content) </td>
```

```
</tr>
```

由于表格通常包含多列，也就是说一行中通常包含多个单元格，所以 tr 元素定义时通常需要包含多个 td 元素，其语法格式可以表示为：

```
<tr>
    <td> 单元格 1 内容(content1) </td> <td> 单元格 2 内容(content2) </td>
    <td> 单元格 3 内容(content3) </td> <td> 单元格 4 内容(content4) </td>
    .....
</tr>
```

例如，下面几行简单的程序就定义了包含 3 个单元格的一行表格内容：

```
<tr>
    <td>Name</td><td>Male/Female</td><td>Born</td>
</tr>
```

其中 3 个单元格的内容分别是“Name”、“Male/Female”、“Born”。

6.7.6 table 元素

table 元素与 tr 元素、td 元素一起，可用来创建能容纳文本和图像的表格，并可设置表格各列中文本和图像的对齐方式。table 元素用于定义表格的列结构，但并不定义列及列间距的宽度。实际显示时，WAP 手机的浏览器将根据表格的宽度，尽可能地显示出该表格；如有必要，还会适当地调整各列的宽度，及列内文本及图像的换行。利用 table 元素的不同属性，我们可指定表格的列数、列内文本及图像的对齐方式等。table 元素可以使用单独的 <table/> 标签来定义，也可以使用成对的 <table> 与 </table> 标签进行定义。其语法格式如下：

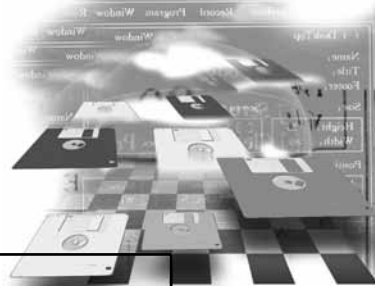
```
<table align="alignment" title="label" columns="n"/>
```

或

```
<table align="alignment" title="label" columns="n">
    内容(content)
</table>
```

其中各个属性的功能和用法介绍如下：

(1) align。该属性用于指定表格各列中文本和图像的对齐方式，共有 L、C 和 R 三种选



择。其中，“L”是指文本和图像向当前列的左边界对齐，“C”是指文本和图像向当前列的中间竖线对齐，“R”是指文本和图像向当前列的右边界对齐。默认值为 L，即左对齐方式。

(2) title。该属性用于指定 table 元素的标题，主要用来描述当前表格。

(3) columns。该属性用于指定表格的列数，该值不能为 0，必须取大于 0 的整数。

下面我们来分析一个建立表格的 WML 例子，程序如下：

```
<wml>
  <card id="card1" title="Weather Forecast">
    <p>
      <table columns="3">

        <tr>
          <td>Day</td><td>Wthr</td><td>Temp</td>
        </tr>

        <tr>
          <td>M 6/7</td><td></td><td>25' C</td>
        </tr>

        <tr>
          <td>T 6/8</td><td></td><td>27'
C</td>
        </tr>

        <tr>
          <td>W 6/9</td><td></td><td>24' C</td>
        </tr>

        <tr>
          <td>T 6/10</td><td></td><td>28' C</td>
        </tr>

        <tr>
          <td>F 6/11</td><td></td><td>29' C</td>
        </tr>
      </table>
    </p>
  </card>
</wml>
```

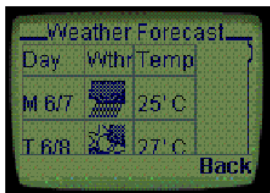


图 6.14 table 元素使用举例

```
</card>
</wml>
```

可以看到，该表格由 6 行 3 列组成，其中第 2 列的内容都是图像。该程序的显示效果如图 6.14 所示。

下面再给出一个简单表格的例子，大家可以使用 WAP 手机浏览一下该程序，看看表格的实际样子。

```
<wml>
  <card>
    <p align="center">
      <i>Hello</i><br/>
      <b><i>World!</i></b>
      <table title="mytable" align="right" columns="2">
        <tr>
          <td>1-1</td> <td>1-2</td>
        </tr>
        <tr>
          <td>2-1</td> <td>2-2</td>
        </tr>
      </table>
    </p>
  </card>
</wml>
```

本章小结

本章以 WML 的元素和事件为主线，详细讲解了 WML 的编程方法。内容涉及卡片与卡片组处理元素、任务类元素、事件及其相关元素、变量设置元素与变量设置的有关规定、用户输入处理元素、文本格式化元素以及锚、图像、定时器方面的处理元素等。本章内容都是与 WML 编程有关的十分重要的内容，希望大家都够熟练地掌握这些内容。

为便于大家系统地掌握 WML 元素的使用方法，我们这里集中给出各种元素的名称及语法格式，如表 6.3 所示。



表6.3 WML元素及其语法格式一览表

元 素	功 能	语 法 格 式
a	a 元素是 anchor 元素的简化形式, 它内嵌了 anchor 元素需要包含的 go 元素功能来完成超链接定位, 并且不再包含其他任何变量设置	<pre> href="href" 文本(text) </pre> 或: <pre> 文本(text) </pre>
access	access 元素是由一个单独的标签即<access>标签实现的元素, 用于定义 WML 整个卡片组的操作权限, 即访问控制参数	<pre><head> <access domain="domain" path="path"/ > ... </head></pre>
anchor	anchor 元素用于创建一个超链接的头部, 超链接的其余部分为用户指定的 URL 地址。当程序运行中用户选中超链接时, 浏览器即会被引入到超链接指定的地址, 如其他卡片组或同一卡片组中的其他卡片	<pre><anchor title="label" > 任务(task) 文本(text) </anchor></pre>
b	使用粗体字显示文本	<pre> 文本(text) </pre>
big	使用大字体显示文本	<pre><big> 文本(text) </big></pre>
br	br 元素用于插入一个回车符, 并开始一个新行	<pre>
</pre>
card	card 元素用于定义一个卡片的各种属性及相关内容等	<pre><card id="name" title="label" newcontext="boolean" ordered="true" onenterforward="href" onenterbackward="href" ontimer="href"> 内容(content) </card></pre>
do	do 元素提供了一个通用的事件处理机制, 使得用户可以参与当前卡片的事件处理	<pre><do type="type" label="label" name="name" optional="boolean"> 任务(task) </do></pre>
em	指定增强显示文本	<pre> 文本(text) </pre>
fieldset	fieldset 元素用于设置输入框和相应的说明文本, 从而用户就可以利用 input 元素等借助该输入框输入所需的数据信息	<pre><fieldset title="label" > 内容(content) </fieldset></pre>
go	go 元素主要用来定义浏览器需要导航的 URL 地址	<pre><go href="href" sendreferer="false true" method="get post" accept-charset="charset"> 内容(content) </go></pre>
head	head 元素用于指定卡片组的头, 即与卡片组整体有关的信息, 包括 meta 数据和 access 控制信息	<pre><head> 内容(content) </head></pre>
i	使用斜体字显示文本	<pre><i> 文本(text) </i></pre>

续表

元 素	功 能	语 法 格 式
img	img 元素用于在格式化的文本中放置和显示一	<pre><img alt="text" src="url" localsrc="icon"</pre>

	幅图像	<code>align="alignment" height="n" width="n" vspace="n" hspace="n" /></code>
input	input 元素用于定义文本实体对象, 包含有对输入文本内容的格式、数据类型、长度、值、变量名等多种属性的具体规定	<code><input name="variable" title="label" type="type" value="value" default="default" format="specifier" emptyok="false true" size="n" maxlength="n" tabindex="n"/></code>
meta	meta 元素用于定义 WML 卡片组相关的通用 meta 信息	<code><meta name="name" http-equiv="name" content="value" forua="true false" scheme="format"/></code>
noop	noop 元素表示什么也不做, 是一个空操作	<code><noop/></code>
onenterbackward	当用户使用 prev 任务或类似的任务来导航至某一卡片时, onenterbackward 事件才可发生。换句话说, 当用户从历史堆栈中选取 URL 地址, 并通过浏览器打开这一地址对应的卡片时, onenterbackward 事件才能发生	<code><card id="name" title="label" newcontext="boolean" ordered="true" onenterforward="href" onenterbackward="href" ontimer="href" ></code> 内容(content) <code></card></code> 或: <code><template onenterforward="href" onenterbackward="href" ontimer="href" ></code> 内容(content) <code></template></code> 或: <code><onevent type="onenterbackward"></code> <code><go href="href"/></code> 或其他任务(task) <code></onevent></code>
onenterforward	onenterforward 事件仅当用户使用 go 任务或类似于 go 任务的任务来定位和浏览卡片时才可发生。设置 onenterforward 事件后, 当用户进入当前卡片组时, 浏览器就会定位 onenterforward 属性或<go/>标签中指定超链(href)的 URL 地址, 并打开 URL 指定的卡片	<code><card id="name" title="label" newcontext="boolean" ordered="true" onenterforward="href" onenterbackward="href" ontimer="href" ></code> 内容(content) <code></card></code> 或: <code><template onenterforward="href" onenterbackward="href" ontimer="href" ></code> 内容(content) <code></template></code> 或: <code><onevent type="onenterforward"></code> <code><go href="href"/></code> 或其他任务(task) <code></onevent></code>

续表

元 素	功 能	语法格式
onevent	onevent 元素通过<onevent>和</onevent>标签可以把包含的任务(task)与特定的事件捆绑在一起。当用户激活这一特定事件时,onevent 元素所绑定的任务就会被立即执行	<pre><onevent type="type"> 任务(task) </onevent></pre>
onpick	onpick 事件在定义时一般通过 onpick 属性指定一些项目,当用户选择或取消这些项目时,即可触发 onpick 事件,执行 onpick 属性所指定的项目	<pre><option value="value" onpick="href"> 内容(content) </option></pre>
ontimer	ontimer 用于指定一个事件,当<timer/>标签指定的时间到期后,浏览器就执行 ontimer 所指定的这个事件	<pre><card id="name" title="label" newcontext="boolean" ordered="true" onenterforward="href" onenterbackward="href" ontimer="href" > 内容(content) </card> 或: <template onenterforward="href" onenterbackward="href" ontimer="href" > 内容(content) </template></pre>
optgroup	optgroup 元素用于将多个相关的 option 元素进行分组,用户浏览器可以借助这种分组来安排选项列表的显示布局,以方便用户选择	<pre><optgroup title="label" > 内容(content) </optgroup></pre>
option	option 元素用于定义 select 元素中的一组单选题,并可包括事件和单选题的显示文本等信息	<pre><option title="label" value="value" onpick="href" > 内容(content) </option></pre>
p	p 元素用于划分段落,使当前文本换行并插入一个空白行	<pre><p align="alignment" mode="wrapmode" > 文本(text) </p> 或<p align="alignment" mode="wrapmode" /></pre>
postfield	postfield 元素用于指定当浏览器接到 URL 请求时,向源服务器(origin server)传送的域名及域值	<pre><postfield name="name" value="value"/></pre>
prev	prev 元素用于指定将浏览器导航至历史堆栈中的前一个 URL 地址	<pre><prev/> 或 <prev> 内容(content) </prev></pre>
refresh	refresh 用于刷新当前的卡片,对卡片内指定的变量进行更新	<pre><refresh> 内容(content) </refresh></pre>
select	select 元素允许用户从选项列表中选择所需的项目	<pre><select title="label" multiple="false true" name="variable" default="default" iname="index_var" ivalue="default" tabindex="n" > 内容(content) </select></pre>

续表

元 素	功 能	语 法 格 式
setvar	setvar 元素用于指定在当前上下文内容中的变量的值，从侧面影响正在运行的任务	<setvar name="name" value="value" />
small	使用小字体显示文本	<small> 文本(text) </small>
strong	进一步加强文本的增强显示	 文本(text)
table	table 元素与 tr 元素、td 元素一起，可用来创建能容纳文本和图像的表格，并可设置表格各列中文本和图像的对齐方式	<table align="alignment" title="label" columns="n"/> 或 <table align="alignment" title="label" columns="n"> 内容(content) </table>
td	td 元素用于规定表格单元格的内容	<td> 单元格内容(content) </td>
template	template 元素用于为当前卡片组中的所有卡片定义一个模板，统一规定卡片的某些参数	<template onenterforward="href" onenterbackward="href" ontimer="href" > 内容(content) </template>
timer	timer 元素用于设定一个定时器，可以延时显示卡片组、卡片，或实现 WML 程序的等待操作，或在卡片组和卡片之间实现切换以取得动画效果	<timer name="variable" value="value" />
tr	tr 元素用于定义表格的行	<tr> <td> 单元格内容(content) </td> </tr>
u	显示文本时增加下划线	<u> 文本(text) </u>
wml	wml 元素用于定义一个卡片组，并通过<wml>与</wml>标签包含和封装该卡片组中的所有卡片及信息	<wml xml:lang="lang" > 内容(content) </wml>



第 7 章 WMLScript 语法基础

WMLScript 是属于无线应用协议 WAP 应用层的一部分,使用它可以向 WML 卡片组和卡片中添加客户端的处理逻辑,目前最新的版本是 1.1 版。WMLScript 1.1 是在欧洲计算机制造商协会制定的 ECMAScript 脚本语言的基础上,经过修改和优化而制定的。它能够更好地支持诸如移动电话类的窄带宽通信设备,在 WML 编程中使用 WMLScript 可以有效地增强客户端应用的灵活性,而且,我们也将 WMLScript 作为一个单独的工具使用,开发出功能强大的 WAP 网络应用和无线网页。本章我们将详细讲解 WMLScript 1.1 编程的基础语法知识,如基本规则、变量与数据类型、操作符与表达式等。为了叙述上的简便,以后我们将“WMLScript 1.1”简称为“WMLScript”。

7.1 简单例子：WML 程序中调用 WMLScript 函数

经过前两章的学习,熟悉 C 语言的读者可能会认识到,WML 的函数功能、逻辑运算功能等都是十分有限的。而 WMLScript 提供了丰富的函数功能,我们在 WAP 应用开发中可以使用 WMLScript 来增强 WML 编程,调用 WMLScript 语句和函数的命令可以直接写在 WML 程序中并实现相应的功能。因此,WMLScript 成为扩展 WML 编程能力的主要开发工具。为了使大家对 WMLScript 有一个总体印象,我们这里先举出一个简单的例子,说明在 WML 程序中可以加入和调用 WMLScript 的语句或函数。程序如下:

```
<wml>
  <card id="confirmation">
    <do type="accept" label="Calculate">
      <go href="calculateTotal.wmls#calculateTotal($(price),$(unit),$(ship))"/>
    </do>
  <p>
```

```
        Each is $$ $(price),  
        You ordered $(unit) copies,  
        Shipping is $$ $(ship), and  
        Total is $$ $(total)  
    </p>  
</card>  
</wml>
```

该 WML 程序中有一句 go 元素语句：`<go href="calculateTotal.wmls#calculateTotal($($price),$($unit),$($ship))"/>`，它建立了一个对 WMLScript 外部函数“calculateTotal(\$(\$price),\$(\$unit),\$(\$ship))”的调用，而含有 calculateTotal 函数的 WMLScript 程序名为 calculateTotal.wmls，它位于当前 WML 程序所在的同一目录中。程序内容如下：

```
extern function calculateTotal(a,b,c)  
{  
    //a: unit price  
    //b: number of units  
    //c: shipping charges  
  
    var total=a*b+c;  
    WMLBrowser.setVar("total",total);  
    WMLBrowser.refresh( );  
};
```

根据前两章的知识，我们知道 WML 自身没有提供用于数据计算和统计的语句，但现在我们通过 WMLScript 函数，很容易地实现了这一点，丰富了 WML 程序的功能。这样，本例的 WML 程序就可以让 WAP 手机用户实现无线网上的电子商务活动，实现 WAP 手机的网上购物和付款。当然，这里有关在 WML 程序中调用 WMLScript 的方法，以及 WMLScript 外部函数的编写方法我们并没有给出解释。随着后面 WMLScript 内容的深入介绍，大家会逐步了解和掌握这些方法的。

7.2 WMLScript 的主要优点及其字节码解释器

WMLScript 具有一套定义好的字节码和一个解释器参考结构。无线网络传输中，



WMLScript 的数据均以二进制格式进行传输,所以,用户可以使用窄带宽通信信道,从而能够保持客户端手机只需要最小限度的内存。ECMAScript 修改后得到的 WMLScript 能够更快、更小、更容易地编译程序为字节码形式。所有这些特点,使得 WMLScript 具备了 WML 所不能具备的很多优点和功能。

7.2.1 使用 WMLScript 的主要优点

WMLScript 的设计宗旨是为 WAP 系统提供一般的脚本处理能力,使用 WMLScript 我们可以进一步补充基于 XML 的 WML 语言的编程功能,开发针对窄带宽的网络应用及内容,如文本、图像、选择列表等,我们可以使用简单的格式编写出更灵活和更具可读性的用户界面。WMLScript 具备的 WML 所不能具备的优点和功能主要包括如下几个方面:

- (1) 检查用户输入的合法性;
- (2) 扩展用户浏览器的功能,比如允许程序员开发手机的电话呼叫、发送短消息、存储电话号码、管理电话簿或 SIM 卡等;
- (3) 生成用户端的确认、提示、警告信息或操作对话框,并使之快速显示在浏览器上;
- (4) 在用户浏览器更改后,能够对浏览器端的软件和参数进行扩展与配置;
- (5) 最大程度地克服客户端的窄带宽通信连接限制,并提供丰富的程序功能;
- (6) 补充 WML 并使之实现针对微型移动终端设备的多种服务,如支持高级用户界面、增加客户端智能性、提供用户浏览器外围功能的访问能力,以及在服务器与客户端浏览传输数据时减少带宽占用等。

7.2.2 WMLScript 的字节码解释器

在 WMLScript 的字节码解释器解释之前,WMLScript 语言编写的文本格式的程序将被首先编译为二进制格式的代码。编译时,编译器通常先将 WMLScript 程序分成若干个编译单位,每个单位的程序都包含一定数量的语句行和 WMLScript 函数,然后,WMLScript 的编译器将按照这些编译单位,逐一将 WMLScript 程序编码成 WMLScript 的字节码。编译器将每个单位的 WMLScript 程序作为输入内容,而把对应的字节码作为输出内容。当用户通过 WAP 手机调用 WMLScript 程序时,编译器的编码功能即被激活、执行。图 7.1 给出了 WMLScript 解释器的一般结构。

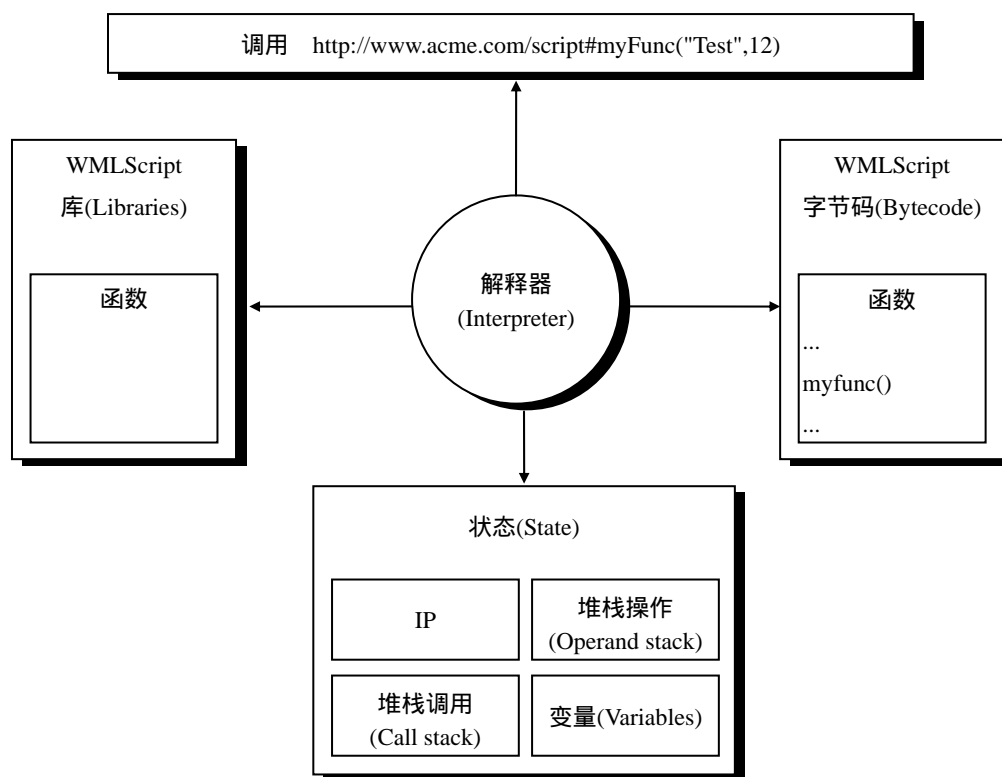
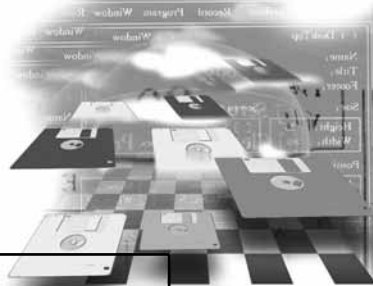


图 7.1 WMLScript 解释器的一般结构

从图中可以看出，当我们调用某个无线网页或应用时，WMLScript 解释器将调用和执行与之相应的字节码形式的 WMLScript 函数，每个函数通常会含有一些参数或指令，必要时解释器就从 WMLScript 库中获取这些函数的相应解释，以备执行函数功能时使用。WMLScript 函数在解释过程中，解释器将生成并维持相应的状态信息，主要包括以下 4 个方面：

- (1) IP 地址。它代表了指令指针，用于指定需要解释为字节码形式的函数或其他指令地址；
- (2) 变量(Variables)。即指当前函数的各种参数和变量，包括运行中的中间变量；
- (3) 堆栈操作(Operand stack)。用于解决调用函数和调用者之间的表达、争端和所需的数据与指令处理；
- (4) 堆栈调用(Call stack)。WMLScript 函数在解释过程中还可以调用其他的函数或访问库函数，这一任务主要由堆栈调用功能来解决，它能处理函数调用和访问中的各种参数，并可返回所需的地址及结果。



在 WMLScript 中,函数调用仅指从 WMLScript 编译单位中调用外部函数,外部函数不属于 WMLScript 库函数,通常是由开发者建立或第三方提供的函数,但外部函数中可以包含 WMLScript 库中的函数。

此外,默认状态下,WMLScript 解释器对所有字符串的操作都只能使用一种字符集,通常是手机用户的母语字符集,这可通过相应的 WMLScript 库函数设置进行指定。规定好母语字符集后,解释器将据此对其他字符集的字符串进行编码转换,以满足手机浏览器的需要。

7.3 WMLScript 基本规则

WMLScript 在许多基本规则方面沿用了 WML 的做法。不过,由于 WMLScript 是以 C 语言为蓝本而制定的,所以它的语法特征和 C 语言非常相像。如果大家对 C 语言比较熟悉,那么学习和掌握这部分内容应当是比较容易的。

7.3.1 WMLScript 与 URL

与 WML 一样,WMLScript 也沿用了 WWW 和 HTML 访问资源的 URL、HTTP 等规范,并扩大了 URL 使用的范围。在 WMLScript 中,不仅超链接、文件路径及文件名可以作为 URL 处理,外部函数、访问控制信息等也可作为 URL 处理。

为此,WMLScript 采用了 WML 的变通方法,即改进 HTML 命名资源位置的方式,采用程序段锚点(Fragment Anchor)的形式来处理资源定位。程序段锚点根据文档 URL 规则进行定义,并按照程序段标识符前加井字号(#)的方式书写。使用程序段锚点,WMLScript 程序可以在 WMLScript 编译单位内定位任一指定的函数,并可在调用该函数的同时传递所需的参数。

例如,WAP 手机用户通过浏览器调用外部 WMLScript 函数时,可先写出该函数所在的 URL 地址,如 `http://www.acme.com/myScripts.scr`;然后,将函数名及参数作为程序段锚点处理,比如 `testFunc('Test%20argument',-8)`,这样最后含有程序段锚点的 URL 地址书写格式为:`http://www.acme.com/myScripts.scr#testFunc('Test%20argument',-8)`。注意其中加上了井字号(#)。

浏览器接到这样一个调用后,将首先执行访问控制检查。如果调用者没有访问权限,



则终止该函数的调用。否则，继续执行调用操作，将锚点中的函数名与编译程序中的外部函数进行匹配，如果不合适，则终止执行；否则，就进一步对函数参数的格式、个数、类型、参数值属性等进行判断，如果符合要求，则函数调用成功，否则拒绝函数调用。

7.3.2 词法结构

WMLScript 编程中的词法结构并不复杂，我们下面就从大小写敏感、空格、换行、注释及保留字等方面讲解相关的具体规则。

(1) 内容类型。WMLScript 的内容类型主要针对文本形式和二进制形式两种情况，类型结构可以在服务器端进行指定，具体形式为：

文本形式：text/vnd.wap.wmlscript；

二进制形式：application/vnd.wap.wmlscriptc。

具体指定方法我们在第 4 章已经介绍过，这里不再重述。

(2) 大小写敏感。WMLScript 1.1 是一种大小写敏感的脚本语言。它所涉及的各种关键字、变量和函数名都必须合理地使用大小写。

(3) 空格和换行。一般情况下，WMLScript 程序执行时将忽略所有的空格、制表符和换行符等。但如果把这些特殊字符通过代码进行表述，或者作为字符串进行处理时，WMLScript 将不再忽略它们。例如，字符串"Oct 28, 2001"中含有空格，该空格在执行时就不会被忽略，它与不含空格的字符串"Oct28, 2001"是不同的。

(4) 注释。与 WML 编程一样，在 WMLScript 脚本程序中也可以加入注释内容。注释内容不被程序执行，且注释不能嵌套。WMLScript 的注释方法有两种：

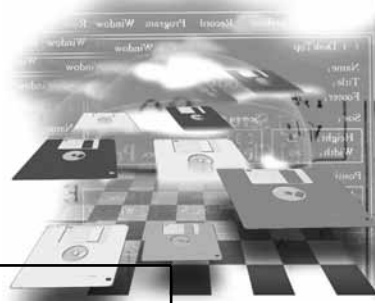
其一，行注释。即使用双斜杠号(//)引导一行内容，这一行内容全部作为注释内容。如：

//这是一行注释，由双斜杠号开始到这行结束都是注释。

其二，块注释。即以符号“/*”开始，而以符号“*/”结束的其间所有内容都是注释内容。如：

/* 这就是块注释，夹在中间的内容就是注释内容 */

(5) 数据类型与直接编码。WMLScript 允许将 4 种类型的数据直接编码并可嵌入在程序之中。直接编码的 4 种数据类型为：整数、浮点数、字符串和布尔型。另外，“无效型”值



也可直接编码。


整数。当以十进制、十六进制或八进制方式使用整数时，可对这类整数进行直接编码。例如：- 98、0xE5、032 等。

编程书写时，十进制的数字均不以 0 开头，只包含 0~9 的数字串；十六进制的数均以 0X 或者 0x 开头，只包含 0~9、a~f 或者 A~F 的字符串；八进制的数均以 0 开头，只包含 0~7 的数字串。

浮点数。浮点数通常定义为含有小数点的数字，可以包含小数和指数部分。浮点数的形式较多，可以是十进制的整数或浮点数，可以是分数，也可以是指数；但一个浮点数必须至少有一个数。

指数是以 e 或 E 开头，后面跟着一个整数。指数是以 10 为底的幂。例如：e0 是 10 的零次幂，e-2 是 10 的负 2 次幂即等于 0.01。指数可以带符号，正号(+)或者减号(-)，它们分别代表是正指数和负指数。

例如：5.69 可以表示成以下几种形式的浮点数：5.69、5.69e0、5.69E0、.569E1 或 569e-2。
再如，以下形式的数都表示一个浮点数：十进制整数 5、6、9 等，十进制浮点数 5.69、56.9、.569 等，分数 1/3、5/9、5/6 等，指数 e0、e5、e-6 等。

 注意，如果编程时在 WMLScript 程序中使用了超出定义范围的浮点数，则程序执行时将会导致编译错误。浮点数溢出时，WMLScript 将该数作为 0 处理。

字符串。字符串是指定义在成对的双引号(" ")或单引号(' ')之间的内容。例如：'Welcome to WAPnet !'、"27 September, 2000 09:05 PM"、"36% off retail"等都是合法的字符串。在 WMLScript 编程中，字符串其实是一个指向字符串所在位置的内存指针。程序运行中我们可以修改字符串的值，甚至可以对字符串进行所需的运算。

由于 WMLScript 只允许使用成对的双引号或单引号来定义字符串，所以程序中使用一个单引号或一个双引号时就会出现编译错误。

考虑到有些特殊字符不能在字符串中直接显现出来，所以 WMLScript 提供了转义序列来表示这些特殊字符。表 7.1 给出了这些特殊字符及其转义序列。

表 7.1 特殊字符及其转义序列

符号名称	符 号	转义序列	Unicode
撇号或单引号	'	\'	\u0027
双引号	"	\"	\u0022

续表

符号名称	符 号	转义序列	Unicode
反斜杠	\	\\	\u005C
斜杠	/	\/	\u002F
退格		\b	\u0008
送纸		\f	\u000C
换行		\n	\u000A
回车		\r	\u000D
水平制表符		\t	\u0009
由 2 个十六进制数 hh 指定的字符(符合 Latin-1 ISO8859-1 标准)		\xhh	
由 3 个八进制数 ooo 指定的字符(符合 Latin-1 ISO8859-1 标准)		\ooo	
由 4 个十六进制数 hhhh 指定的 Unicode 字符		\uhhhh	

例如：“Example”、‘Specials: \x00 \’ \b’或“Quote: \””等都属于包含有转义序列的合法字符串。其中第 2 个字符串含有\x00 \’ \b，第 3 个含有\”。

布尔型。它只有 true 和 false 两个数值，用于表示 WMLScript 中的“真值”或“假值”。布尔型数据可参与异、或等运算，具体规则我们后面介绍。

无效型。也称为“空类型”，它是 WMLScript 支持的一个表示无效值的量，以 invalid 表示。该量与 C 语言中的 NULL 类似。例如，下面的程序中就是一条判断 invalid 的语句：

```
var x = 12;
var y = 0;
if ((x/y) == invalid)
{
    错误：除数为零！
};
```

(6) 保留字。WMLScript 中定义有一个保留字集合，含有一些表示特殊意义的单词，这些词不能另外定义，也不能作为其他标识符。WMLScript 中的保留字如下：

access	http	agent	if	break	isvalid	continue	meta	header
div	name	div=	path	domain	return	else	typeof	while
equiv	url	extern	use	for	user	function	var	

另外，WMLScript 还为将来的版本预留了一些保留字，主要有：

case	finally	catch	import	class	private	const	public	debugger
------	---------	-------	--------	-------	---------	-------	--------	----------

sizeof default struct do super enum switch export throw
extends try

WMLScript 还有一些没有使用的保留字：

delete null in this lib void new with

(7) 标识符。WMLScript 的标识符可以指定或命名 3 种元素：变量、函数和标注。标识符不能以数字开头，但能以短下划线(_)开头，而且，标识符不能是 WMLScript 的保留字。例如，timeOfDay、speed、quality、HOME_ADDRESS、_myName、____、var0 等都是合法的标识符；而以数字或非短下划线的特殊符号开头的字串，以及保留字等都属于不合法的标识符，如 while、for、if、my~name、\$sys、123、3pieces、take.this 等。

由于 WMLScript 是严格区分大小写的，所以字母相同但大小写不同的标识符不是同一个标识，例如，Work 和 work 就是不同的标识符。

(8) 名称空间。WMLScript 提供了比较自由的名称空间，同一标识符可以同时用作不同的目的。例如，作为某一函数名称的标识符，还可以同时用作变量名、函数参数、程序标注等，使用时它们的属性或值等并不相互影响。在下面的简单的例程中，myTest 这一标识符既用作了函数名，又用作了变量名、函数参数名、常量名。显然，WMLScript 的这一特点为我们编写程序提供了很大的方便。例程如下：

```
use url myTest "http://www.acme.com/script";

function myTest(myTest) {
    var value = myTest#myTest(myTest);
    return value;
};
```

7.3.3 WMLScript 程序的基本书写规则

由上面的简单例程，我们可以看出 WMLScript 程序的基本书写规则：

- (1) 程序由若干语句或函数组成，函数又由若干语句组成；
- (2) 每个完整的语句后面必须加上分号(;)，语句关键词与操作数之间必须有空格；分号(;)是 WMLScript 程序的组成部分；



(3) 函数体之间必须使用成对的花括号({ })括起来,而且函数结束时在右花括号(})的后面还要加上分号(;) ;函数说明部分,如函数名、函数类型、函数参数等要放在花括号({ })的前面;

(4) 有些语句可能也需要使用花括号({ })包含内容,这类语句通常也可以放在函数中,所以花括号({ })是可以嵌套的,例如:

```
function mybookCheck(givenbook) {  
    if (givenbook > 20) {  
        var newbook = givenbook + 100;  
    } else {  
        newbook = 100 - givenbook;  
    };  
    return newbook;  
};
```

当然,不同的语句、参数、变量等元素在声明和书写时可能还有一些更细的要求,具体我们后面介绍这些元素时再专门给出。

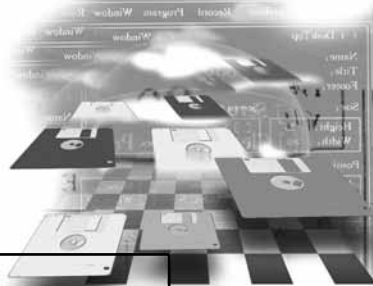
7.4 变量与数据类型

变量及数据类型是所有编程语言的重要概念和组成部分,WMLScript 对此也不例外。它对其变量使用方法和数据类型定义方法给出了详细的规定。变量通常与某数据值相对应,我们可以给变量赋值,并可在程序执行中改变变量的值。下面我们讲解 WMLScript 有关变量与数据类型的详细规定。

7.4.1 变量及其声明

变量是在 WMLScript 脚本程序中具有值的符号名,或说标识符。使用变量可以存储和改变程序中需要的数据。与 C 语言不同的是,WMLScript 仅支持函数内定义的变量或用于传送数值的参变量。

变量使用前必须进行声明,也就是定义变量,即指定变量的名字。声明变量的关键字是 var,它的后面跟上作为变量名的合法的标识符,并于最后加上分号(;),即完成一个变



量的声明。声明变量时可以使用 `var` 一次声明多个变量名，相邻变量名之间使用逗号(,)间隔。语法格式为：

```
var 变量名 1, 变量名 2, ....., 变量名 n ;
```

例如，以下几行语句都是合法的声明变量的语句：

```
var x;  
var price5d;  
var x, y, z;
```


WMLScript 还允许我们在定义变量的同时对变量进行初始化。方法是使用等号(=)直接在变量右边赋值，语法格式为：

```
var 变量名 = 初始值 ;
```

例如：

```
var size = 12;
```

如果声明变量不进行初始化，那么 WMLScript 将自动把该变量初始化为空字符串("")。

需要说明的是，同一函数中定义的所有变量的变量名必须是唯一的，否则就会导致编译错误。

一般情况下，我们在给变量命名的时候，都希望能够使用有意义的变量名。例如，当需要使用一个变量表示一本书的价格时，虽然将变量命名为 `j` 或 `book` 都没有什么错误，但若能命名为 `bookPrice` 则会使得 WMLScript 脚本程序具有更好的可读性，可以方便编程人员进行脚本的编写和调试工作。而且，由于 WMLScript 在给变量命名时不能使用保留字，所以考虑到避免由于一时的疏忽使变量名与保留字发生冲突，我们建议使用多个单词组合在一起作为一个变量的名称，这是一个比较好的解决办法。例如，如果要定义一个变量来存储一本书的价格，那我们可以不妨使用 `bookPrice` 或者 `book_price` 作为变量的名称，这样，一方面可以使变量的定义显得更加清晰，另一方面也可以避免变量与保留字的冲突问题。

以上只是我们对于变量命名的建议，并不是强制性的要求，用户完全可以不按照我们的要求来做，只要遵守 WMLScript 对于标识符命名的要求就可以了，但养成良好的编程风格不论是对编程人员还是对脚本编写人员来说，都是十分有意义的。

7.4.2 变量的作用域与生命期

一个变量的作用域是指在程序中能够引用这个变量的一段代码。由于 WMLScript 仅支持函数内定义的变量，所以 WMLScript 变量的作用域通常就是定义它的那个函数。在该函数之外，变量不再发挥直接作用。

例如，下面的 priceCheck() 函数中有 newPrice 和 givenPrice 两个变量，其中 givenPrice 是为函数传递参数值的参变量。这两个变量只在 priceCheck() 函数中可以发挥作用，也就是说，它们的作用域是 priceCheck() 函数：

```
function priceCheck(givenPrice) {  
    if (givenPrice > 100) {  
        var newPrice = givenPrice;  
    } else {  
        newPrice = 100;  
    };  
    return newPrice;  
};
```

变量的生命期是指从变量声明开始到失效为止。变量的生命期也被称为变量的持久期、存活期。任一个变量在定义它的整个函数内都是有效的，函数内的任何语句块都不会削减变量的生命期或限制变量的作用域。

如果一个变量未经声明就直接使用，或声明过后再次声明，都会破坏变量的生命期。前一种情况会导致变量没有开始生命期，即没有“生命”；而后一种情况则导致变量生命期没有结束以前就重新赋予生命期，即让它多次“降生”。这都会导致变量无效使用。下面函数中的变量使用就说明了这一问题：

```
function foo() {  
    x = 1; // 错误：变量使用前没有声明，该变量还没有“生命”。  
    var x,y,z;  
    y = x + 3;  
    var zd = invalid;  
    if (x) {  
        var (y); // 错误：这一变量已经声明，这里是重复声明。
```




```
};  
};
```

7.4.3 变量的使用

WMLScript 的变量只能在定义它的函数内使用。使用时需要声明变量，声明变量时可以同时对变量赋值，甚至对变量进行运算。例如，下面的简单函数就说明了变量的这种灵活的使用方法：

```
function ourAge() {  
    var myAge = 38;  
    var yourAge = 26;  
    var ourAge = myAge + yourAge;  
    return ourAge;  
};
```

使用变量时可通过调用变量名字的形式来实现。上面例子中的“var ourAge = myAge + yourAge;”一句，通过调用变量名，变量 ourAge 对变量 myAge 和变量 yourAge 实行了求和操作。


7.4.4 变量类型与数据类型

WMLScript 是一种“弱类型”的语言，即其变量没有确定的类型。WMLScript 变量的类型由该变量所赋数据的类型决定，并根据数据类型的改变而改变。WMLScript 只支持内部定义的数据，因此我们编写程序时无需指定 WMLScript 变量的类型，WMLScript 将根据变量所赋数据的类型自动进行匹配。由于 WMLScript 的数据类型共有整数、浮点数、字符串、布尔型和“无效型”五种类型，所以 WMLScript 变量的类型所能匹配的也就是这五种类型。

例如，下面几条语句定义了几个变量并赋予了初始值，由于这些作为初始值的数据都有确定的类型，所以变量的类型相应地就由这些数据的类型所决定：

```
var flag = true;           // 定义变量 flag 并赋初值 true，这是布尔型的数据，所以变量为布尔型
```

```
var number = 22;           // 定义变量 number 并赋初值 22，数据为整型，所以变量也为整型
var temperature = 36.159;  // 数据为浮点型，所以变量也为浮点型
number = 2 * temperature;  // 由于 temperature 为浮点型，所以运算后 number 成为浮点型
number = "XI";             // 程序中又改变了变量 number 的值，被赋予了一个字符串，
                           // 所以此时 number 变量的类型为字符串型
var except = invalid;      // 数据为无效值，所以变量也为无效型
```

 与变量相对应，还有一种在程序运行中不改变的量，即常量。常量就是可以直接使用的字符串类型、数值类型等数据类型的数据，例如"hello"、3 等。虽然 WMLScript 不强调常量，但我们应当了解有常量这么回事。

7.4.5 变量值域

由于变量类型由其所赋数据的类型决定，所以变量值域与其所赋数据的可取值范围是等价的。下面我们就给出整数、浮点数、字符串和布尔型数据的取值范围，以参照确定相应类型变量的值域。

(1) 整数的范围。WMLScript 支持的整数是 32 位的，也就是说整数的取值范围是从 -2147483648 到 +2147483647。我们可以在程序运行期使用 Lang 函数来取得这些值，如：

```
Lang.maxInt(); // 获取最大的整数
Lang.minInt(); // 获取最小的整数
```

(2) 浮点数的范围。它是指以 WMLScript 浮点数的精度所能表示的最小和最大数值。WMLScript 支持 32 位的单精度浮点数，其最大值是 3.40282347E+38，最小的非零的数值是 1.17549435E-38 或更小(按照正常的精度)。

我们可以使用浮点 Float 函数库在程序运行期取得这些数值：

```
Float.maxFloat(); // 获取 WMLScript 所支持的最大浮点值
Float.minFloat(); // 获取 WMLScript 所支持的最小浮点值
```

对于运行期出现的一些特殊的浮点数，WMLScript 将按照下述规则处理：

其一，如果操作结果是一个不能被单精度浮点数所能表示的数值，那么该结果将被认



为是 invalid，即无效值；

其二，如果操作结果发生下溢出，那么结果将作为 0.0 处理；

其三，负的零和正的零是完全相等的。

(3) 字符串的范围。任何由字母、数字或特殊字符组成的符号串都是 WMLScript 合法的字符串。我们可以使用字符串来初始化字符串变量，也可以使用 WMLScript 中定义的有关字符串的操作或 String 库中的函数来控制字符串。例如，下面几行语句都是对字符串的有效定义和操作：

```
var msg = "Hello";  
var len = String.length(msg);  
msg = msg + 'Worlds!';
```

(4) 布尔型数据的范围。布尔型数据只有 true 和 false 两个取值，这也是布尔型变量的两种取值。我们可以使用布尔型数据去初始化或指定某一变量的数值，或将布尔型变量写入一个需要布尔值做为参数的语句。布尔值可以是数值运算的结果，也可以是逻辑运算的结果。

下面就是定义布尔型变量并赋初值的例句：

```
var truth = true;  
var lie = !truth;
```

7.5 操作符与表达式

在 WMLScript 中，表达式可以把变量、常量与操作符结合起来，经过运算能够产生一定的运算结果。表达式运算后产生的结果可以是整数型、浮点型、字符串型或布尔型的数据。其实，对于表达式我们并不陌生，例如， $1+2$ 就是一个简单的表达式。

WMLScript 的表达式主要有两种类型。一种是赋值表达式，即把数据赋给变量的一种表达式，例如， $myBook=3$ ，在这个表达式中，将 3 赋给变量 myBook，同时，这个表达式本身也有一个运算结果，那就是 3。另外一种为运算表达式，它是只运算产生一个运算结果而不进行赋值操作的表达式，例如 $1+2$ 就是一个运算表达式，在这个表达式运算产生的结

果是 3，但这个表达式并没有把运算结果赋给变量。

在表达式运算的过程中，表达式中操作一个或者两个数据产生运算结果的符号称作操作符，被操作符操作的数据称作操作数，在 WMLScript 中我们会使用到各种操作符，下面就对操作符及有关的表达式进行详细讲解。

7.5.1 赋值操作符

赋值操作符用于赋值操作，即给变量指定所需的数值，它能把右操作数的运算结果赋给左操作数，最简单的赋值操作符就是“=”，例如 $x=2$ ，就是将 2 赋值给变量 x 。再如以下几行语句都是赋值操作：

```
var a = "abc";  
var b = a;  
b = "def";
```

赋值操作符不需要指定使用对象，也不会改变赋值操作符右边变量的数值。WMLScript 的赋值操作符主要包括以下几种：

(1) =。用于赋值操作，将右操作数赋给左操作数。

(2) +=。将右操作数与左操作数进行相加运算，然后把运算结果赋值给左操作数。例如，假设 $x=3$ ，那么 $x+=2$ 运算后的结果为 $x=5$ 。

+= 是比较特别的操作符，因为它可以将两个字符串相连，所以 += 操作符也可以对字符串进行操作，然后将连接后的字符串赋给左操作数。例如，假设 $x="Happy "$ ，那么 $x+="New Year"$ 运算后结果是 $x="Happy New Year"$ 。

(3) -=。将左操作数减去右操作数，然后把运算结果赋值给左操作数。例如，假设 $x=3$ ，那么 $x-=2$ 运算后的结果为 $x=1$ 。

(4) *=。将左操作数与右操作数进行相乘运算，然后把运算结果赋值给左操作数。例如，假设 $x=3$ ，那么 $x*=2$ 运算后的结果为 $x=6$ 。

(5) /=。用右操作数除以左操作数，然后把运算结果即求得的商赋值给左操作数。例如，假设 $x=6$ ，那么 $x/=2$ 运算后的结果为 $x=3$ 。

(6) div=。用右操作数除以左操作数，然后把运算结果中的整数部分赋值给左操作数。例如，假设 $x=7$ ，那么 $x \text{ div } 2$ 运算后的结果为 $x=3$ 。

(7) %=。功能是求余数并赋值，用右操作数除以左操作数，最后把运算得到的余数赋值



给左操作数。例如，假设 $x=7$ ，那么 $x\%=3$ 运算后的结果为 $x=1$ 。

(8) $\ll=$ 。功能是带符号左位移并赋值，即将左操作数和右操作数进行左位移操作，再将结果赋给左操作数。例如，假设 $x=6$ ，那么 $x\ll=2$ 运算后的结果为 $x=24$ 。

(9) $\gg=$ 。可将左操作数和右操作数进行右位移操作，再将结果赋给左操作数。

(10) $\gg\gg=$ 。用于将左操作数和右操作数进行补零右位移操作，再将结果赋给左操作数。

(11) $\&=$ 。功能是先将左操作数和右操作数进行位与的逻辑运算，然后再将结果赋给左操作数。例如，假设 $x=6$ ，那么 $x\&=3$ 运算后的结果为 $x=2$ 。

(12) $\^=$ 。功能是先将左操作数和右操作数进行位异或的逻辑运算，然后再将结果赋给左操作数。

(13) $|=$ 。先将左操作数和右操作数进行位或的逻辑运算，然后再将结果赋给左操作数。

7.5.2 数学运算操作符

数学运算操作符可以对数值类型的操作数进行运算，然后返回一个数值类型的运算结果。例如：


```
var y = 1/4;           // 4 除 1 得 0.25，所以 y 被赋值 0.25
```

```
var x = y*3+(++b);     // 这是一个更为复杂的数学运算
```

WMLScript 的数学运算操作符介绍如下：

(1) $+$ 。这是加运算操作符，它对应着数学运算中的加法运算，例如表达式 $1+2$ 的运算结果为 3。

加操作符还可以对字符串类型的操作数进行运算，然后将两个字符串相连起来作为运算结果。例如，`"How are "+"you!"` 的运算结果为 `"How are you!"`。

 注意，这里讲到的“ $+$ ”和前面介绍的“ $+=$ ”操作符都可用于字符串的运算，其他有关字符串的操作功能全部在 String 函数库中。该库情况我们下一章会详细介绍的。

(2) $-$ 。即减操作符，对应着数学运算中的减法运算，例如表达式 $2-1$ 的运算结果为 1。

同时，“ $-$ ”还是一个取负操作符，当它作为取负操作符的时候，它只有一个操作数，取负操作符的功能是返回操作数的相反数。例如如果变量 j 的值为 2，那么 $-j$ 这个表达式的值为 -2。

(3) $*$ 。这是乘操作符，它对应着数学运算中的乘法运算，例如表达式 $2*3$ 的运算结果

为 6。

(4) /。即除操作符，对应着数学运算中的除法运算，但 WMLScript 中的除法运算有些特别，在 WMLScript 中，除法运算后的结果是一个浮点数，而不象 C 语言或者 Java 语言那样在整数进行除法运算时将运算结果强行转化整数。在 WMLScript 中， $1/2=0.5$ ，而在 Java 中， $1/2=0$ 。

(5) div。这是整除操作符，对应着数学运算中的整除运算，运算后的结果是一个整数，这一点与 C 语言或者 Java 语言中的情况是一样的，可以在整数进行除法运算时将运算结果强行转化整数。例如， $5 \text{ div } 2=2$ 。

(6) %。即取模操作符，它对应着数学运算中的取模运算，也就是将两个操作数相除，返回相除后的余数，例如表达式 $5\%3$ 的运算结果为 2。

取模操作符主要用于判断一个数字是否能被另外一个数字整除，例如，如果我们希望知道某个年份是否是闰年，我们可以采用如下判断方法：如果这个年份能够被 4 整除，并且不能被 100 整除，那么，这个年份是闰年，另外，如果这个年份能够被 400 整除，那么这个年份也是闰年。假设年份为 theYear，那么判断是否为闰年的 WMLScript 表达式如下所示：

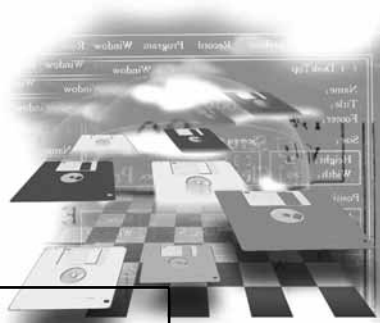
```
((theYear%4==0)&&(theYear%100!=0))||(theYear%400==0)
```

其中 && 代表的是逻辑运算中的与运算，|| 代表的是逻辑运算中的或运算，关于逻辑运算，我们后面会详细介绍。

(7) ++。这是递增操作符，它只有一个操作数，操作数可以在操作符的左边，也可以在操作符的右边，它所完成的运算操作是将操作数加 1。假设操作数名称为 j，值为 2，那么 ++j 是首先将 j 加 1，然后返回 j 的值 3；而 j++ 则是先返回 j 的值 2，然后将 j 加 1。

在循环中，我们常常会用到递增操作符，虽然 $x=x+1$ 和 $x++$ 在运算后对于 x 的效果是完全相同的，但在大多数情况下，我们会选择使用 $x++$ 。

(8) --。这是递减操作符，它与递增操作符的作用正好相反。递减操作符完成的运算操作是将操作数减 1。例如，假设操作数名称为 j，值为 2，那么 --j 先将 j 减 1，然后返回 j 的值 1；而 j-- 是先返回 j 的值 2，然后将 j 减 1。



7.5.3 位操作符

位操作符在运算时先将操作数转化为 32 位的二进制数，然后对每个操作数分别按位进行运算，运算后再将二进制的结果按照标准 WMLScript 数值类型返回运算结果。例如，对于操作数 6，二进制的表示方式为 110，而对于操作数 3，二进制的表示方式为 11。位操作符共有 7 种，分别介绍如下。

(1) &。这是位与操作符，它可以对两个操作数按位进行与操作，其运算规则是：

$0 \& 0 = 0$ ， $0 \& 1 = 0$ ， $1 \& 0 = 0$ ， $1 \& 1 = 1$

例如，在进行 $6 \& 3$ 运算时，将 6 和 3 按二进制方式表达出来，就是 110&011，然后再逐位按位进行与操作：左起第 1 位 $1 \& 0$ 、左起第 2 位 $1 \& 1$ 、左起第 3 位 $0 \& 1$ ，从而最后结果为 010，这是二进制表示的 2，所以 $6 \& 3 = 2$ 。

(2) |。这是位或操作符，它可以对两个操作数按位进行或操作，运算规则是：

$0 | 0 = 0$ ， $0 | 1 = 1$ ， $1 | 0 = 1$ ， $1 | 1 = 1$

例如，在进行 $6 | 3$ 运算时，按照二进制方式表达就是 110|011，逐位按位进行或操作后结果为 111，所以 $6 | 3 = 7$ 。

(3) ^。这是位异或操作符，它可以对两个操作数按位进行异或操作，其运算规则是：

$0 \wedge 0 = 0$ ， $0 \wedge 1 = 1$ ， $1 \wedge 0 = 1$ ， $1 \wedge 1 = 0$

例如，在进行 $6 \wedge 3$ 运算时，按照二进制方式表达就是 110^011，按位进行异或操作后结果为 101，所以 $6 \wedge 3 = 5$ 。

(4) ~。这是位非操作符，它只有一个操作数，可对操作数按位进行非操作，运算规则是： $\sim 0 = 1$ ， $\sim 1 = 0$ 。

例如，在进行 ~ 6 运算时，WMLScript 中 6 的二进制表达式实际上应该是 00000000 00000000 00000110，在前面介绍几个位操作中，因为前面的 0 的与、或、异或操作结果都是 0，不影响运算结果，所以我们把这些 0 给忽略了，但在位非操作时，我们必须保留这些 0，按位进行非操作后，结果为 11111111 11111111 11111111 11111001，它的数值为 -7，所以 $\sim 6 = -7$ 。

(5) <<。这是左移操作符，它可以对左操作数进行向左移位操作，右操作数给定了要移动的位数，在移位过程中，左操作数的最低位补充 0。

例如，对于表达式 $6 << 1$ ，6 的二进制表达方式为 110，然后向左移动 1 位，移位结果为



1100，所以 $6 \ll 1 = 12$ 。

(6) \gg 。这是右移操作符，它可以对左操作数进行向右移位操作，右操作数给定了要移动的位数，在移位的过程中，丢弃向右移出的位。

例如，对于表达式 $6 \gg 2$ ，6 的二进制表达方式为 110，然后向右移动 2 位，移位结果为 1，所以 $6 \gg 2 = 1$ 。

(7) \ggg 。这是填 0 右移操作符，它与右移操作符相似。当对正整数进行操作时，它们的效果完全相同；不同之处在于，当进行负整数右移操作时，因为负数转化为二进制后，最高位为 1，所以在进行右移操作后，最高位仍然补充 1，而在进行填 0 右移操作时，最高位补充的是 0，因此，这时负数将转化为正数。

例如： $-3 \ggg 1 = -2$ ， $-3 \ggg 1 = 2147483646$ 。这是因为，-3 的二进制表示为 11111111 11111111 11111111 11111101，在右移操作后为 11111111 11111111 11111111 11111110，而在进行完填 0 右移操作后为 01111111 11111111 11111111 11111110。

7.5.4 逻辑操作符

逻辑操作符可以将布尔类型的表达式组合起来，完成逻辑运算操作，然后返回逻辑运算的结果——真或假，这样就可以完成比较复杂的逻辑判断工作。逻辑操作符共有 3 种： $\&\&$ 、 $\|\$ 和 $!$ ，下面我们就分别介绍一下。

(1) $\&\&$ 。即逻辑与操作符，它只有在两个操作数都为 true 的时候，返回结果为 true，在其他情况下，返回结果为 false 或者 invalid。

例如，表达式 $(3 > 2) \&\& (3 < 4)$ 的逻辑运算结果为 true，因为 $3 > 2$ 的结果为 true，并且 $3 < 4$ 的结果也为 true；而表达式 $(3 > 2) \&\& (3 > 4)$ 的逻辑运算结果为 false，因为这里 $3 > 4$ 的结果为 false，从而导致整个表达式的结果为 false。

(2) $\|\$ 。这是或操作符，它在两个操作数至少有一个为 true 的时候，返回结果为 true，而当两个操作数都为 false 时，返回结果为 false 或者 invalid。

例如，表达式 $(2 > 3) \|\ (3 > 4)$ 的逻辑运算结果为 false，因为 $2 > 3$ 的结果为 false，并且 $3 > 4$ 的结果也为 false；而表达式 $(2 < 3) \|\ (3 < 4)$ 的逻辑运算结果为 true，因为这里 $3 < 4$ 的结果为 true，从而导致整个表达式的结果为 true。

(3) $!$ 。即非操作符，它只有一个操作数。当操作数为 true 时，返回结果为 false；当操




作数为 false 时，返回结果为 true。

例如，表达式 $!(2>3)$ 逻辑运算结果为 true，因为 $2>3$ 的结果为 false，而表达式 $!(2<3)$ 的逻辑运算结果为 false，因为 $2<3$ 的结果为 true。

我们给出下面一条语句，或许能够更形象地表达逻辑操作符的运算规则：

```
weAgree = (iAmRight && youAreRight) || (!iAmRight && !youAreRight);
```

由于 WMLScript 只能使用布尔值进行逻辑运算，所以当对其他类型的数据进行逻辑运算时，需要先这些类型的数据自动转换为布尔型。具体转换规则我们后面讨论。

需要说明的是，如果与、或操作符的第一个操作数是 invalid，那么 WMLScript 将不计算第二个操作数的值，并且整个结果就是 invalid。下面的例句说明了这一点：

```
var a = (1/0) || foo(); // 由于 1/0 为 invalid，所以结果为 invalid，不再调用函数 foo()
var b = true || (1/0); // 结果为 true
var c = false || (1/0); // 结果为 invalid
```

7.5.5 比较操作符

比较操作符可以把操作数进行比较，然后返回一个逻辑值，表明这个比较操作的结果是否为真。比较操作符的操作数可以是数值类型或者字符串类型的数据。比较操作符也被称为关系运算符。例如，下面两条语句就包含有比较操作符：

```
var res = (myAmount > yourAmount);
var val = ((1/0) == invalid); // val = invalid
```

WMLScript 支持的比较操作符共有 6 种，下面就分别介绍一下。

(1) ==。即等于操作符，它可以比较两个操作数是否相等。如果两个操作数相等，则返回 true，否则返回 false。

例如，对于 $3==3.0$ 的比较，返回结果为 true；而对于 $3==4$ 的比较，返回结果为 false。

(2) !=。即不等操作符，它可以比较两个操作数是否相等。如果两个操作数相等，则返回 false，否则返回 true。

例如，对于 $3!=3.0$ ，返回结果为 false；对于 $3!=4$ ，返回结果为 true。

(3) >。即大于操作符。其运算规则是，如果左操作数大于右操作数，则返回 true，否则

返回 false。

例如，对于 $3>2$ ，返回结果为 true；对于 $3>3$ ，返回结果为 false。

(4) $<$ 。即小于操作符。运算规则是：如果左操作数小于右操作数，则返回 true，否则返回 false。

例如，对于 $3<4$ ，返回结果为 true；对于 $3<3$ ，返回结果为 false。

(5) $>=$ 。这是大于等于操作符。运算规则是：如果左操作数大于或等于右操作数，则返回 true，否则返回 false。

例如，对于 $3>=2$ ，返回结果为 true；对于 $3>=3$ ，返回结果为 true；对于 $3>=4$ ，返回结果为 false。

(6) $<=$ 。这是小于等于操作符。运算规则是：如果左操作数小于或等于右操作数，则返回 true，否则返回 false。

例如，对于 $3<=4$ ，返回结果为 true；对于 $3<=3$ ，返回结果为 true；对于 $3<=2$ ，返回结果为 false。

除了数值类型之外，字符串类型也可以进行比较，在比较字符串的过程中，WMLScript 会把字符串中的每个字母转换成相应的 ASCII 码值，然后从第一个字符开始比较两个字符串中相应的字符，根据它们的 ASCII 码值进行判断。

例如，`"hello"=="Hello"` 的返回结果为 false，这是因为左操作数的第一个字符为 h，右操作数的第一个字符为 H，h 的 ASCII 码值为 104，H 的 ASCII 码值为 72，从第一个字符开始就不相等了，所以表达式的值为 false。

再如，这两个表达式：`"hello">"Hello"` 和 `"hello">="Hello"`，它们的返回结果都为 true。这是因为左操作数第一个字符的 ASCII 码值为 104，而右操作数的第一个字符的 ASCII 码值为 72， $104>72$ (当然同样 $104>=72$)，所以两个表达式的值都为 true。

而如果表达式为 `"came">="come"`，那么如何比较呢？首先，比较两个操作数的第一个字符，都是 c。这样，不能直接判定两个操作数的大小。所以接下来，要比较两个操作数的第二个字符，左操作数为 a，右操作数为 o，a 的 ASCII 码值为 97，o 的 ASCII 码值为 111，当然 $97<111$ ，所以对于这个表达式来说，它的返回结果为 false。

在比较字符串中，我们使用得最多的一种情况是判断某个字符串类型的变量是否为空串，假设，我们判断的变量名称为 theMessage，那么对于 `theMessage==""` 这个表达式来说，如果 theMessage 为空串，则表达式为 true，否则为 false。



根据以上的分析和讨论，我们可以比较运算的基本规则：

- (1) 对于整型数据，按照整数规则进行比较；
- (2) 对于浮点型数据，按照浮点数规则进行比较；
- (3) 对于字符串型数据，按照顺序比较字符串中的字符，比较时以字符的 ASCII 码值为准进行判断；
- (4) 对于布尔型数据，true 比 false 大；
- (5) 对于无效型数据 Invalid，如果只要有一个操作数是 invalid，那么整个比较就是 invalid。

在比较的过程中，不同类型的数据可能会隐含地发生数据类型转换，具体转换规则我们后面介绍。

7.5.6 其他几种操作符

除了前面介绍的几类操作符，WMLScript 还提供了几种比较特殊的操作符，我们这里分别介绍一下。

(1) 数组操作符。WMLScript 数组操作符所操作的并不是真正的数组，而是字符串。而且，WMLScript 不支持普通的数组。WMLScript 标准的 String 库内的函数都将字符串看作是一个数组，并按照模仿数组的行为对其进行操作，字符串中每一个字符按照从左至右的顺序进行编号，该编号可以像数组下标一样使用。例如：

```
function dummy() {  
    var str = "Mary had a little lamb";  
    var word = String.elementAt (str,4,"");  
};
```

其中 String.elementAt (str,4,"")就把字符串变量 str 看作数组进行处理，有关该函数的处理规则我们将在下一章 String 库函数部分介绍，这里暂不展开。

(2) 逗号操作符。该操作符可以将多个表达式连接在一起，形成一个表达式。例如：

```
for (a=1, b=100; a < 10; a++,b++) {  
    ... 其他语句或函数...  
};
```

其中的“a=1, b=100;”和“a++,b++”中都使用了逗号操作符，分别完成了两个表达式的运算。逗号操作符实际上相当于多个表达式的组合，比如“a=1, b=100;”相当于“a=1;”和“b=100;”两个表达式的组合。

在为声明的变量赋值的逗号，或函数调用参数之间的逗号都不是逗号操作符。一般来说，逗号操作符都要使用括号保护起来，而且逗号操作符运算的结果是第二个操作数的结果。例如：

```
var a = 2;
var b = 3, c = 5;      // 其中的逗号不是逗号操作符
var b = 3, c = (a, 3); // 其中(a, 3)中的逗号是逗号操作符，它运算的结果是第二个操作数，即 c=3
myFunction("Name", 3*(b* a, c)); // 参数"Name"后面紧邻的逗号不是逗号操作符，而 3*(b* a, c)
                                // 中的逗号是逗号操作符，其运算结果是取 c 的值，由于上面
                                // 已有 c=3，所以 3*(b* a, c)最后的结果是 3*3=9
```

(3) 条件操作符。条件操作符使用的一般形式是 `:(condition)?val1:val2`，它是 WMLScript 中唯一的一个三目操作符，有三个操作数，能够根据布尔运算的结果给一个变量赋值。

条件操作符的运算规则是：在 condition 为 true 的时候，返回结果为 val1；在 condition 为 false 的时候，返回结果为 val2。例如，如果一个人的年龄大于等于 18 岁，我们就认为他是一个成人(adult)，否则我们就认为他是一个未成年人(youth)，这个判断操作可以使用条件操作符来实现：`(age>=18)?"adult":"youth"`。

条件操作符其实是个 if-then-else 语句，执行原则是如果 condition 为 true，则取 val1，否则取 val2。

(4) typeof 操作符。由于 WMLScript 支持的数据类型有整数型、浮点型、字符串型、布尔型和无效型，所以表达式的判断类型也不外这么几种。有时我们需要判断表达式的类型，这时就可以使用 typeof 操作符来操作。该操作符可以通过返回一个整数来表示给定表达式的类型。其运算规则是：表达式为整数型，则返回 0；表达式为浮点型，则返回 1；表达式为字符串型，则返回 2；表达式为布尔型，则返回 3；表达式为无效型，则返回 4。

typeof 操作符的返回结果只有整数型，它不会再将结果自动转换为其他数据类型。

例如，假设 `str = "123"`，现求 `mytype = typeof str`。由于 str 为字符串，所以表达式类型



为字符串型，故 `myType = 2`。

(5) `isvalid` 操作符。该操作符用于检测给定表达式的有效性。如果表达式是有效的，那么返回值就是 `true`，否则就是 `false`。`isvalid` 操作符的返回结果只有布尔型，它也不会再将结果自动转换为其他数据类型。

例如，`str = "123"`，是个字符串，`str` 的表达式是有效的，所以 `ok = isvalid str` 的结果是 `true`。再如，表达式 `1/0` 是无效的，所以 `isvalid (1/0)` 的运算结果是 `false`，所以 `tst = isvalid (1/0)` 的结果是 `false`。

7.5.7 表达式

表达式通常由操作符、常量、数值函数或变量等组成，本节开头我们已经介绍了表达式，这里不再重述。

我们想要强调的是，WMLScript 支持大多数其他编程语言都支持的表达式。因此，即便一个常量或者变量也可以作为表达式，例如：`:357`、`58.23`、`"This is too simple"`、`'This works too'`、`true`、`myAccountB` 等都是表达式。而且，更为复杂的表达式可以使用简单的表达式、操作符和函数调用等来组成。例如，`myAccount + 3`、`(a + b) / 3`、`initialValue + nextValue(myValues)` 等都是合法的表达式。

7.6 数据类型自动转换规则

前文述及，WMLScript 在进行某些运算时需要对数据进行类型转换。比如进行逻辑运算时，就需要把其他类型的数据自动转换为布尔型数据，然后才能进行逻辑运算。转换时，WMLScript 能够根据当前操作符的特性要求，自动匹配所需的数据类型并自动进行转换。本节我们就详细讨论与此相关的转换规则。

7.6.1 一般转换规则

在默认状态下，WMLScript 的操作符可以对符合数据类型要求的操作数进行直接操作，如果操作数的类型不符合要求，则尝试着对操作数的类型进行转换。然而，并不是所有的数据类型都可以互相转换，有些特殊类型是无法转换的，此时 WMLScript 就会给出无效值



或错误信息作为处理结果。

WMLScript 进行数据类型转换的一般规则如下：

(1) 转换为字符串型。除无效型即 `invalid` 之外，其他 3 种类型的数据都可以转换为字符串型的数据：

其一，整数型数据转换时将按照十进制数的规则，把整数作为字符串。例如，整数 567 转换时，将变为字符串 "567"。

其二，浮点型数据转换时将直接把数据转换为字符串形式，且保持它所代表的值不变。比如，3.14 转换为字符串时为 "3.14"，该字符串所代表的值仍为 3.14 或 .314e1 等。

其三，布尔型的值 `true` 转换时将变成字符串 "true"，布尔型值 `false` 将转换成字符串 "false"。

(2) 转换为整数型。除了无效型即 `invalid` 以及浮点数不能转换为整数型数据外，其他两种数据类型都有可能转换为整数：

其一，如果字符串所包含的字符全部都是数字，则该字符串可以转换为整数。如果前几位为 0，则转换时一并忽略。例如，字符串 "2345" 转换时会变为整数 2345，字符串 "0023045" 转换时会转换为整数 23045。

其二，布尔型的值 `true` 转换时将变成整数值 1，布尔型值 `false` 转换时将变为整数值 0。

(3) 转换为浮点型。除了无效型即 `invalid` 不能转换为浮点型数据外，其他 3 种数据类型都有可能转换为浮点型数：

其一，如果字符串中包含有一个合法的浮点型数，那么该字符串就可以转换为浮点型数。例如，字符串 "2345.789" 转换时会变为浮点数 2345.789，字符串 "0023.045" 转换时会转换为浮点数 23.045；而字符串 "23W45.789" 不含有合法的浮点数，所以它不能转换。

其二，整数可以转换为相等的浮点数。例如，整数 234 转换时就变为浮点数 234.0。

其三，布尔型的值 `true` 转换时将变成浮点型数值 1.0，布尔型值 `false` 转换时将变为浮点型数值 0.0。

(4) 转换为布尔型。除了无效型即 `invalid` 不能转换为布尔型数据外，其他 3 种数据类型都有可能转换为布尔型数：

其一，空字符串 ("") 可以转换为布尔型值 `false`，其他所有字符串均转换为布尔型值 `true`。

其二，整数值 0 可以转换为布尔型值 `false`，其他所有整数均转换为布尔型值 `true`。

其三，浮点型数值 0.0 可以转换为布尔型值 `false`，其他所有浮点型数均转换为布尔型值



true。

(5) 转换为无效型。凡不符合上述转换规则的数据类型，其数据在转换时都转换为无效型，即得到 invalid 的值。另外，如果操作符的操作数就是或含有 invalid，那么操作的结果一般都是 invalid。

7.6.2 操作符数据类型转换规则

前面我们介绍的是数据类型转换的一般规则，操作符运算时一方面要遵循这些一般规则，另一方面还要遵循一些与逻辑运算相适应的数据类型转换规则。为了便于大家深入了解操作符数据类型的转换方法，我们下面以各种操作数类型为例，说明一般规则之外的一些附加规则。

(1) 操作数类型为布尔型。即操作符要求操作数的数据类型为布尔型，这种情况下，如果操作数是布尔型或者是能够被转换为布尔型的数据，那么当前操作符就可以进行布尔运算并返回布尔值，否则一律返回 invalid。

例如 `true && 3.4`、`1 && 0`、`"A" || ""` 或 `!42` 等都可以返回布尔值，而 `!invalid` 与 `3 && invalid` 则都会返回 invalid。

(2) 操作数类型为整数型。即操作符要求操作数的数据类型为整数型，这种情况下，如果操作数是整型或者是能够被转换为整型的数，那么就可进行整数运算并返回整数值；否则就会返回 invalid。

比如，`"7" << 2`、`true << 2` 都会返回一个整数值；而 `7.2 >> 3`、`2.1 div 4` 则会返回 invalid。

(3) 操作数类型为浮点型。这种情况下，如果操作数是浮点数或者是能被转换为浮点型的数，那么当前操作符就可以进行浮点运算并返回浮点值，否则就会返回 invalid。

(4) 操作数类型为字符串型。此时，如果操作数是字符串类型或者是能够被转换为字符串类型的数据，那么就可进行字符串运算并返回字符串，否则就会返回 invalid。

(5) 操作数类型为一目的整数或浮点数。即操作符只要求一个操作数，该操作数需要为整数或浮点数，此时，如果操作数是整数或可以被转换成整数型数值，那么操作符就可进行整数转换并返回整数值，否则返回 invalid。例如，`+10`、`-"33"` 都将返回整数值，`+true` 返回整数值 1，`-false` 返回整数值 0，等等。

如果操作数是浮点数或者是能被转换为浮点型的数，那么当前操作符将进行浮点运算并返回浮点值。比如，`-10.3`、`+"47.3"` 等都可返回浮点数值，而 `-"ABC"`、`-"9e9999"` 因不含有

合法的浮点数而无法转换，所以返回 invalid。

(6) 操作数类型为整数或浮点数(二目)。由于这种情况下有不只一个操作数，所以根据操作数类型的不同有 3 种处理方法：

如果有一个操作数是浮点数，则就将另外一个数转换为浮点数，并进行浮点运算和返回浮点值。例如， $100/10.3$ 、 $3.4*4.3$ 、 $"2.3"*3$ 最后返回的都是浮点数。

如果操作数是整数或者是能够被转换为整型的数，那么就会进行整数运算并返回整数值。如 $33*44$ 、 $"10"*3$ 、 $"10"-"2"$ 的运算结果都是返回一个整数值。

如果操作数都是浮点数，或都能被转换为浮点型的数，那么就会进行浮点运算并返回浮点值。

上述 3 种情况如果不能成立，则运算结果就会返回 invalid。如 $3.2*"A"$ 、 $.9*"9e999"$ 以及 $invalid*1$ 运算后各自返回的都是 invalid。

(7) 操作数为整数、浮点数或字符串。这种情况是操作符进行的比较复杂的运算，共有 3 种情况：

如果操作数是整数或者是能够被转换为整型的数，那么操作符就进行整数运算并返回整数值。比如 $12+3$ 、 $3<false$ 的结果都是整数值。

如果有一个操作数是浮点数，那么运算时就将另外的操作数均转换为浮点数，进行浮点运算并返回浮点值。例如， $32.4+65$ 、 $43.2<77$ 、 $9.9+true$ 都是按照浮点数处理，最后返回的是浮点数值。

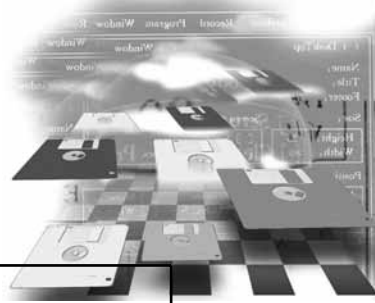
如果有一个操作数是字符串，那么运算时就将另外的操作数均转换为字符串，进行字符串运算并返回字符串。比如 $"12"+5.4$ 、 $"Hey"<56$ 、 $2.7+"4.2"$ 最后返回的值都是字符串。

如果上述 3 种情况都不能成立，则运算结果就会返回 invalid。如 $"A"+invalid$ 返回的就是 invalid。

(8) 操作数为任何类型。如果操作符对任何类型的操作数都承认，那么它可接受任何类型的数据进行运算，如 $=$ 或 $typeof$ 等操作符均属于此类情况： $a = 37.3$ 可以接受并返回浮点数， $b = typeof "s"$ 可以接受并返回字符串，等等。

7.6.3 操作符与数据类型汇总

WMLScript 操作符的操作数有的只需要一种数据类型，有的可以有多种数据类型。前



第7章 WML Script 语法基础

一种情况一般不会涉及操作数类型的数据转换，但后一种情况往往需要类型转换。为了方便大家从总体上把握这些操作符及其操作数的类型转换规则，我们将它们列在表 7.2 和表 7.3 中，供大家参考。

表7.2 单一类型操作数的操作符及其规则

操作符	操作数类型	结果类型	执行操作
!	布尔型	布尔型	一目逻辑非
&&	布尔型	布尔型	逻辑与
	布尔型	布尔型	逻辑或
~	整数型	整数型	一目位非
<<	整数型	整数型	左位移
>>	整数型	整数型	带符号右位移
>>>	整数型	整数型	补零右位移
&	整数型	整数型	位与
^	整数型	整数型	位异或
	整数型	整数型	位或
%	整数型	整数型	求余
div	整数型	整数型	整除
<<=, >>=, >>>=, &=, ^=, =	第 1 个操作数为变量， 第 2 个操作数为整数型	整数型	位操作赋值
%=, div=	第 1 个操作数为变量， 第 2 个操作数为整数型	整数型	数学运算操作赋值

表7.3 多种类型操作数的操作符及其规则

操作符	操作数类型	结果类型	执行操作
++	整数型 或 浮点型	整数型/浮点型	一目的前增量或后增量
--	整数型 或 浮点型	整数型/浮点型	一目的前减量或后减量
+	整数型 或 浮点型	整数型/浮点型	一目加
-	整数型 或 浮点型	整数型/浮点型	一目减
*	整数型 或 浮点型	整数型/浮点型	相乘
/	整数型 或 浮点型	浮点型	相除
-	整数型 或 浮点型	整数型/浮点型	相减法
+	整数型, 浮点型 或 字符串型	整数型/浮点型/字符串型	相加 或 字符串型连接
<, <=	整数型, 浮点型 或 字符串型	布尔型	小于, 小于等于
>, >=	整数型, 浮点型 或 字符串型	布尔型	大于, 大于等于
==	整数型, 浮点型 或 字符串型	布尔型	同值相等
!=	整数型, 浮点型 或 字符串型	布尔型	异值不等
*=, -=	第 1 个操作数为变量, 第 2 个操作数为整数型 或 浮点型	整数型/浮点型	数学操作符赋值
/=	第 1 个操作数为变量, 第 2 个操作数为整数型 或 浮点型	浮点型	相除赋值

续表

操作符	操作数类型	结果类型	执行操作
+=	第 1 个操作数为变量, 第 2 个操作数为整数型、浮点型 或 字符串型	整数型/浮点型/字符串型	数值相加或字符串连接并赋值
typeof	任何类型	整数型	返回内部数据类型(一目)
isvalid	任何类型	布尔型	检查有效性(一目)
?:	第 1 个操作数为布尔型, 第 2、3 个操作数为任何类型	任何类型	条件表达式
=	第 1 个操作数为变量, 第 2 个操作数为任何类型	任何类型	赋值
,	第 1、2 个操作数均可可为任何类型	任何类型	多重判断

本章小结

本章我们先从一个在 WML 程序中调用 WMLScript 函数的简单例子初步认识了 WMLScript 脚本语言, 随后我们介绍了 WMLScript 语言的主要优点、用途及其工作原理, 接下来我们利用较大的篇幅, 详细讲解了 WMLScript 语言的基本语法和规则, 内容涉及 WMLScript 的基本规定、词法结构、程序书写方法、变量与数据类型、操作符与表达式以及数据类型的自动转换规则等。

本章内容都是 WMLScript 脚本语言的基础知识, 大家应当全面掌握。尤其是 WMLScript 的变量、数据类型和操作符等内容, 是我们后面进一步学习 WMLScript 编程的重要准备知识, 学习时大家最好结合我们的举例, 通过例子认识和深入掌握这些知识。

第 8 章 WMLScript 脚本程序设计

8.1 语 句

在 WMI Script 中，每条语句的后面都需要以一个分号(·)结尾。为了养成严谨的编程

WMI Script 语句的书写和排列格式比较自由。我们可以在同一程序行中连续写上多个

WMI Script 的语句主要包括两类。第一类是基本语句,如空语句、表达式语句、块语

8.1.1 基本语句

WMLScript 基本语句主要用于程序格式控制和变量声明，其中有些语句我们已经不太陌生了。

空语句

空语句用于定义一个空的程序行，它没有任何标识符和操作符，也不执行任何操作。它只是以分号(;)结束。其语法格式为：

```
;
```

显然，空语句是一种十分特殊的语句。有时我们为了让程序具有更好的可读性，通常在程序中的适当地方加上几个空语句，以起到分隔或突出的作用。例如，下面的几行程序中就含有一个空语句：

```
str = "Hello!";  
val = 25;  
;  
MyVal = val*val+5;  
alert("Hi, Hi!!!");
```

再如，while 语句用于判断一个条件并在条件满足的时候执行相应的任务，但如果希望条件满足的时候什么也不作，那么就可以给它配上一个空语句，使之条件满足时执行空操作：

```
while ( !poll( device ) );
```

这实际上是 while 语句和空语句组成的两条语句。其中的分号(;)在这里就代表了空语句。这两条语句的作用是在 poll() 函数为真(true)之前一直等待。

表达式语句

表达式语句用于向变量赋值，或进行数学计算，或进行函数调用。表达式语句也是我们最常用的一种语句，语法格式为：

```
表达式;
```



下面几行程序都是合法的表达式语句：

```
str = "Hey " + yourName;
val3 = prevVal + 4;
counter++;
myValue1 = counter, myValue2 = val3;
alert("Watch out!");
retVal = 16*Lang.max(val3,counter);
```

块语句

块语句使用两个花括号({ })包含一个语句集，形成一个语句体。WMLScript 的许多语句都需要使用块语句来实现语句体，块语句的语法格式为：

```
{
    语句列表;
}
```

下面的简单程序就是使用块语句的例子：

```
{
    var i = 0;
    var x = Lang.abs(b);
    popUp("Remember!");
}
```

变量语句

变量语句用于声明变量并可进行变量的初始化赋值。如果用户不赋值，那么 WMLScript 会自动将变量语句声明的变量赋予一个空字符串("")。基本的语法格式为：

```
var 变量名;
```

如果想一次声明多个变量，则相邻变量名之间使用逗号(,)间隔，其语法格式为：

```
var 变量名 1, 变量名 2, ....., 变量名 n;
```

如果想在声明变量时同时初始化变量，则可按如下语法格式书写：

```
var 变量名 = 初始值;
```

为便于大家更好地掌握变量语句，我们给出一个多处使用该语句的程序：

```
function count(str) {  
    var result = 0;           // 声明变量的同时初始化变量  
    while (str != "") {  
        var ind = 0;         // 每次循环都初始化一次  
        // 为退出循环，本块语句内应当提供修改变量 str 值的语句  
    };  
    return result  
};  
  
function example(param) {  
    var a = 0;  
    if (param > a) {  
        var b = a+1;         // 声明 b 变量的同时使用 a 变量来初始化 b 变量  
    } else {  
        var c = a+2;         // 声明 c 变量的同时使用 a 变量来初始化 c 变量  
    };  
    return a;                // 返回 a 变量的值  
};
```

注释语句

严格来讲，注释语句在 WMLScript 中还不算是真正的语句，它只是一种强制性的规定。不过它也有严格的语法和标注方法，所以我们这里还是像其他编程语言处理的一样，把 WMLScript 注释方法以语句的形式介绍一下。

注释在程序执行的时候没有任何作用，但是可以用于对程序进行解释，增强程序的可读性。为了形成良好的编程风格，我们应该养成书写注释的良好习惯，注释有两种表达方式：

(1) 通过双斜线注释一行，这样在双斜线后的字符将成为注释而不被执行。该注释行可以单独一行书写，也可以放在其他语句的后面。

例如，可以进行如下所示的注释：

```
// 变量 j 用于描述每月的天数  
j = 0;    // 我们这里将 j 赋值为 0
```



(2) 通过符号 “/*” 和 “*/” 来规定注释语句，这种注释方式可以进行多行注释，符号 “/*” 和 “*/” 之间的内容就是注释语句。例如，可以进行如下所示的多行注释：

```
/* 我们定义了两个变量：i 和 j。其中：
   i 用于描述每年中的月数，
   而 j 用于描述每月的天数 */
j = 0;      /* 我们这里将 j 赋值为 0 */
```

return 语句

return 语句主要用在函数体中，在函数结束前，可以通过 return 语句，把函数处理的结果返回给调用函数的语句。它的语法格式如下：

```
return 表达式；
```

下面的函数给出了应用 return 语句的例子：

```
function square( x ) {
    if (!(Lang.isFloat(x))) return invalid;
    return x * x;
};
```

8.1.2 条件语句

在条件语句中，当满足某种条件时，就会执行指定的一些代码，而在满足另外某种条件时，则会执行另外一些代码。WMLScript 的条件语句就是 if...else 语句，它的一般表达形式如下：

```
if(条件) {
    代码块 1
}
else {
    代码块 2
}
```



这样，当条件满足时，就执行代码块 1；如果条件不满足则执行代码块 2。代码块 1 和代码块 2 中如果只有一条语句，那么，两边的花括号({ })就可以省略；而如果有多条语句，则必须使用花括号将代码块包括在其中。在 if...else 语句中，其中的 else 部分是可选的，也就是说，我们可以使用如下的表达形式：

```
if (条件){  
    代码块  
}
```

这样，当条件满足时，就执行代码块，如果条件不满足则什么也不做。

例如，如果我们需要对一个学生的成绩(score)进行判定，如果大于等于 60 分，那么我们就认为该学生成绩合格了，反之则认为不合格，同时一并将状态记录到变量 status 中，相应的 WMLScript 语句如下所示：

```
if (score>=60)    status="pass";  
else    status="fail";
```

再如，我们可以通过对天气是否阳光普照(sunShines)的情况进行判断，来给变量 myDay 赋值，并累计好天气(goodDays)的天数。程序如下：

```
if (sunShines) {  
    myDay = "Good";  
    goodDays++;  
} else  
    myDay = "Oh well...";
```

8.1.3 循环语句

使用循环语句可以反复执行某个代码块，直到循环结束条件满足后才停止执行。WMLScript 中有两种循环语句：for 语句和 while 语句，同时还有两种与循环密切相关的操作语句：break 语句和 continue 语句。



for 语句

for 语句可以创建一个带条件的循环，它含有 3 个可选的条件表达式，用于控制循环。这 3 个条件表达式放在一个括号里，并以分号(;)间隔。for 语句的一般语法形式如下：

```
for (初始表达式;循环条件;递增表达式) {  
    代码块  
}
```

for 语句的执行主要包括以下几个步骤：

(1) 执行初始表达式。在一般情况下，初始表达式完成的功能是在循环中对循环计数器赋初值。所以在这种意义上，初始表达式也可以采用“var 变量声明列表;”的形式来定义。

(2) 判断循环条件。如果循环条件为真(true)，则执行循环体中的语句，即至步骤(3)；否则，循环条件为假(false)或为 invalid，就结束循环；

(3) 执行循环代码。然后，再执行递增表达式。一般情况下，我们在递增表达式中对循环计数器进行处理，最后再返回步骤(2)执行。

例如，下面的 for 语句建立了一个循环。初始表达式为定义变量 index 并赋初值 0，循环条件为 $\text{index} < 100$ ，递增表达式为每循环一次 index 增加 1。当 index 增加到 100 的时候，循环结束。程序如下：

```
for (var index = 0; index < 100; index++) {  
    count += index;  
    myFunc(count);  
};
```

while 语句

while 语句也可创建一个循环，它的一般语法表达形式如下：

```
while (循环条件) {  
    代码块  
}
```

while 语句的执行过程包括以下几个步骤：

(1) 判断循环条件是否为真。如果循环条件为真，则执行循环；如果为假或为 invalid，则跳出循环。

(2) 执行循环中的代码块，然后返回步骤(1)。

下面的程序就是使用 while 语句的简单例子：

```
var counter = 0;
var total = 0;
while (counter < 3) {
    counter++;
    total += c;
};
```

其中建立的循环仅当变量 counter 的值小于 3 时执行，否则就结束循环。

显然，如果循环条件不能为假或为 invalid，那么 while 循环就会无休止的反复执行下去。因此，我们在代码块中一定要有能够改变循环条件的变量，否则，就很有可能会陷入死循环而不能终止程序，下面就是一个死循环的例子：

```
var x=1;
var y=0;
while (x<=1) {
    y = x+1;
}
```

在这个程序中，因为变量 x 的值在循环中不能发生变化，所以循环条件在判断的时候永远为真，所以成为了死循环。因此，对于 while 语句我们往往使用如下所示的语法形式：

```
初始表达式
while (循环条件) {
    代码块
    递增表达式
}
```

这种情况下，while 语句的功能和 for 语句的功能就一样了，不过用 while 语句编写的程序可读性更强一些。所以我们可以采用 while 语句来完成 index 增加到 100 的循环。程序



如下：

```
var index = 0;
while (index < 100) {
    count += index;
    myFunc(count);
    index++;
};
```

break 语句

为了更好地解决死循环问题，WMLScript 像大多数编程语言一样提供了 break 语句。break 语句可以使程序执行跳出循环。不论是 for 语句还是 while 语句，只要在循环中使用了 break 语句，那么程序执行到 break 语句后就立即跳出当前循环，然后继续执行下去。

break 语句的语法形式如下：

```
break;
```

例如，在下面的函数中我们使用了 break 语句，它使得当 index=3 时跳出循环。如果不使用该语句，函数中的 while 循环需到 index =6 时才可以结束。程序如下：

```
function testBreak(x) {
    var index = 0;
    while (index < 6) {
        if (index == 3) break;
        index++;
    };
    return index*x;
};
```

continue 语句

continue 语句的功能和 break 语句的功能看起来有些类似，但实际上却不一样。循环执行时遇到 break 语句通常是跳出当前循环，但循环执行到 continue 语句后并不跳出当前循环，而是不执行循环中在 continue 语句后面的代码块，直接结束循环的本轮运行，然后马上开始下一轮循环的运行。

在 while 语句的循环中，遇到 continue 语句后，程序会直接判断循环条件从而开始下一轮循环。在 for 语句的循环中，遇到 continue 语句后程序会直接执行递增表达式，然后判断循环条件从而开始下一轮循环。

例如，我们想利用 for 循环求 1 到 10 之间偶数的和，其 WMLScript 语句如下：

```
var sum=0;
for (var j=1; j<=10; j++) {
    if ( j%2 != 0 )
        continue;
    sum += j;
};
```

在这个例子中，在 $j \% 2 \neq 0$ 的情况下，也就是 j 为奇数的情况下，程序执行 continue 语句，这时，并没有如同 break 语句一样跳出循环的运行，而是不执行循环中后面的语句而直接执行递增表达式开始下一轮循环的执行，这样，就可以不将奇数 j 的值累计入总和中。

再如，我们想利用 while 循环求 0 到 4 之间除 3 以外几个数的和，则可以使用 continue 语句进行控制。程序如下：

```
var index = 0;
var count = 0;
while (index < 5) {
    index++;
    if (index == 3)
        continue;
    count += index;
};
```

这一程序中，当 index 等于 3 时，“ $index == 3$ ”为真，所以执行 continue 语句，不再把此时 index 的值加到 count 中，而是开始下一轮的循环。



8.2 函数的声明与调用

在 WMLScript 中，函数是一种能够完成某种功能的代码块，并可以在脚本中被事件处理或被其他语句调用，也可以被 WML 程序所处理和调用。一般地，当我们编写 WMLScript 脚本时，如果脚本中的代码长度比较长，这时就要根据功能的不同将代码块写入一个或几个函数中；如果函数中的代码长度还是很长，则一般还可以根据功能将函数再进行划分，分成为几个功能更加单一的函数。虽然说这种对长代码的处理方法并不是编写脚本程序的强制性要求，但通过函数的划分和运用，我们可以使得 WMLScript 脚本具有更好的可读性，也便于我们对脚本程序的编写与调试。而且，如果在某些脚本中有多处完全相同的代码块，那么我们也可以将这些代码块写为一个函数，然后在脚本中调用这个函数，从而提高代码的重用性，简化代码的编写工作。

WMLScript 的函数功用和 Java 语言、C/C++ 语言的函数有所不同。我们知道，Java 语言、C/C++ 语言中有函数和过程之分，函数能够完成一定的功能并有返回值，而过程仅可完成一定的功能但没有返回值。可是，WMLScript 中并不区分函数和过程，因为它只有函数，没有过程。WMLScript 的函数完成一定功能后始终有返回值，不过返回值分两种情况，即非空的返回值和空字符串("")形式的返回值。前者是真正的返回值，后者其实相当于没有返回值。也就是说，WMLScript 中的函数同时具有其他语言中的函数和过程的功能。

8.2.1 函数的声明

使用函数时，要根据函数的调用语法来使用，而调用函数前必须声明函数，也就是需要先定义函数。WMLScript 中定义函数的一般方式如下：

```
function 函数名(参数列表)
{
    代码块
};
```

另外，WMLScript 规定使用 `extern` 关键字来声明一个外部函数：

```
extern function 函数名(参数列表)
{
    代码块
};
```

从中可以看出，函数的定义由以下 3 部分组成：

(1) 函数名。即函数的名称，其命名规则应当遵守 WMLScript 的标识符规则(参见 7.3.2 节)。调用函数时都是通过函数名进行调用的，所以函数必须要有函数名。

函数命名时，一般要使用能够描述函数功能的单词来作为函数名，也可以使用多个单词组合进行命名，这样做的好处是能够提高 WMLScript 脚本的可读性。

函数名在同一个 WMLScript 脚本文件里必须是唯一的。如若不然，则会导致函数定义混乱。

(2) 参数列表。即调用函数时需要的参数。参数列表通常是可选的，有的函数需要，有的函数可能不需要。参数列表的作用是向函数传递一些参数，使得函数可以直接使用这些参数的值。

调用函数的时候，参数个数和类型必须和函数定义时所声明的参数个数及类型保持一致。而且函数的参数就如同是函数体内的局部变量，它们在函数调用的时候被初始化。

(3) 代码块。它是函数的主体部分。代码块中的代码包含在一对花括号({ })中，代码块可以执行并完成函数的功能。编写代码块时应当遵循 WMLScript 的编程规则。

有时，函数需要返回一个值给调用函数的语句，则应该在代码块的最后一行使用 return 语句，返回所需的数值。

与 C/C++ 等语言类似，WMLScript 的函数是可以嵌套的，也就是说，在一个函数中还可以调用其他函数。但是，函数声明时不能嵌套，这是 WMLScript 的强制性规定。

下面几行语句就是定义函数的简单例子：

```
function currencyConverter(currency, exchangeRate)
{
    return currency*exchangeRate;
};
```

其中，该函数的名称为 currencyConverter，参数有 currency 和 exchangeRate 两个，函数



代码块包含一条语句，用于返回 currency 和 exchangeRate 的乘积。

下面是一个使用 extern 定义外部函数的例子。其中函数名为 testIt，它没有参数，函数体中定义了两个赋值变量，一个赋整数，一个赋函数值：

```
extern function testIt() {  
    var USD = 10;  
    var FIM = currencyConverter (USD, 5.3);  
};
```

8.2.2 函数的调用

编写好的函数必须经过合法的调用，才可以发挥它应有的作用。函数调用后将返回一个值，比如一个计算结果。WMLScript 中的函数主要可以分为内部函数、外部函数和库函数，下面我们就介绍这 3 类函数的调用方法。

内部函数

所谓内部函数是指函数的定义与其调用函数在同一个脚本文件内的函数，对内部函数的调用称为内部调用。内部函数的调用非常简单，只需提供函数名和所需参数值即可，参数值必须和函数定义时指定的参数个数及类型一致。而且函数调用需要使用操作符来接收或处理被调用函数的返回值。

内部函数可以在其定义之前调用，也可以在其定义之后调用。例如，下面就是一个在函数定义之后调用的例子：

```
function test1 (val) {  
    return val*val;  
};  
  
function test2 (param) {  
    return test1 (param+1);  
};
```

这个例子中定义了两个函数 test1 和 test2。test1 函数用于计算给定参数值的平方并将结果返回；test2 函数将给定的参数值加 1，然后以这个和为参数值，来调用 test1 函数，得到结果后再将该结果返回到调用 test2 函数的语句。



注意，本例中 test2 函数调用了 test1 函数，这种在函数中调用其他函数的方法称为函数调用嵌套。WMLScript 的内部函数、外部函数和库函数都支持嵌套调用，后面我们将专门介绍这方面的内容。

外部函数

外部函数是一个在 WMLScript 外部文件中定义的函数。调用外部函数的方法与调用内部函数的方法基本类似，不同之处在于调用外部函数时一是要指定外部文件的地址及名称，二是要在调用的外部函数名的前面加上外部文件的名称。

WMLScript 规则使用 use url 来指定外部文件，语法格式为：

use url 含有外部函数的外部文件名 外部文件所在的 URL;

这样，WMLScript 的预编译头就可以将外部文件映射为一个可以在内部使用的标识。然后，使用这个标识并加上井号(#)和标准的函数调用即可实现外部函数调用，语法格式为：

外部文件名#外部函数(参数列表);

例如，http://www.host.com/script 下我们需要的外部文件，名为 OtherScript，所以我们可以使用 use url 来指定该文件：

```
use url OtherScript "http://www.host.com/script";
```

这一外部文件中含有我们需要调用的外部函数 testme，则可采用“外部文件名#外部函数(参数列表)”的形式来调用它：

```
OtherScript#testme(param+1);
```

这个例子完整地写出来，就是下面的程序：

```
use url OtherScript "http://www.host.com/script";
```

```
function test(param) {  
    return OtherScript#testme(param+1);  
};
```




库函数

如无特别指明，WMLScript 的库函数一律是指它的标准库函数。因为与标准库函数对应，WMLScript 还有一些非标准的库函数。我们这里先介绍标准库函数，非标准库函数后面再介绍。

所有库函数都有所属的库，函数的库中通常含有一类函数。因此，调用某个库函数时，一要指定它所在的库名，二要指定它的函数名。WMLScript 规定，调用标准库函数时可以通过在函数库的名字后面加上句点号(.)和库函数的标准调用来实现，语法格式为：

函数库名.函数名(参数列表);

例如，WMLScript 的浮点库即 Float 库中有一个开根方的函数 sqrt，该函数只有一个参数，那么调用 sqrt 库函数的方法为：

```
Float.sqrt(number); // 这里要求 number 大于或等于 0
```

下面给出了调用库函数的简单例子。首先以 param 参数值调用 Lang.abs()函数，返回结果加 1 后再作为参数值调用 Float.sqrt()函数，它的返回结果将作为内部函数 test 的返回值：

```
function test(param) {
    return Float.sqrt(Lang.abs(param)+1);
};
```

8.2.3 函数的嵌套调用

WMLScript 的函数定义都是互相平行、独立的，定义函数的时候我们不能在一个函数内定义另外一个函数，也就是说，函数定义是不能嵌套的。但是，函数调用却是可以嵌套的，也就是说，我们可以在调用一个函数的过程中调用另外一个函数。

如图 8.1 所示，就是我们给出的 3 层函数嵌套调用执行过程的示意图。它的执行过程是：

(1) 执行 a 函数开头部分；

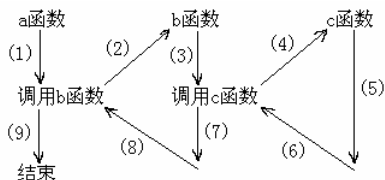


图 8.1 函数的嵌套调用

- (2) 遇到调用 b 函数的操作语句，流程则转去执行 b 函数；
- (3) 执行 b 函数的开头部分；
- (4) 遇到调用 c 函数的操作语句，流程则转去执行 c 函数；
- (5) 执行 c 函数，如果没有其他嵌套的函数，则完成 c 函数的全部操作；
- (6) 返回调用 c 函数的语句，即返回到 b 函数；
- (7) 继续执行 b 函数中尚未执行的操作，直到 b 函数结束；
- (8) 返回 a 函数中调用 b 函数的语句；
- (9) 继续执行 a 函数的剩余操作，直到函数结束。

我们下面给出一个内部函数嵌套调用的例子，其执行过程类似于图 8.1 所示的过程：

```
function myFunC (param1) {  
    return param1*param1- Float.sqrt(Lang.abs(param)+1);  
};  
  
function myFunB (param0) {  
    return myFunC (param0+1)*param0 +12 ;  
};  
  
function myFunA (param) {  
    return myFunB (param*param+1);  
};
```

8.3 预 编 译

WMLScript 的预编译主要用于在编译阶段控制编译器的行为。预编译头一般在文件开头和函数声明之前指定，WMLScript 规定所有的预编译头都是以关键词 `use` 加上指定的预编译属性进行指定。

在大多数的编程中，我们比较常用的预编译行为主要涉及外部文件声明、访问权限和 Meta 信息设置。



8.3.1 外部文件

我们知道，使用 URL 地址可以定位一个 WMLScript 文件。利用该 URL 地址，在 WMLScript 编程中我们可以通过预编译来调用 WMLScript 的外部文件，外部文件预编译头的声明方法是 `use url`，其语法格式如下：

```
use url 外部文件名 "URL 地址"；
```

这样，我们在当前文件的编程中就可以使用该预编译头声明的外部文件，从而可以调用该外部文件里的函数。其语法格式为：

```
外部文件名#函数名(参数列表)；
```

例如，我们希望在当前的 WMLScript 程序中调用 OtherScript 外部文件中的 `check()` 函数，而且我们知道 OtherScript 文件的 URL 地址为 `http://www.host.com/app/script`。因此，我们可以使用 `use url` 来声明这一外部文件：

```
use url OtherScript "http://www.host.com/app/script";
```

随后，我们就可以在程序中调用 OtherScript 中的 `check()` 函数了：

```
function test (par1, par2)
{
    return OtherScript#check (par1-par2);
};
```

其中调用执行的过程如下：

- (1) 找到 WMLScript 外部文件的 URL 地址；
- (2) 当前函数从指定的 URL 地址置装载外部文件；
- (3) 检测外部文件的内容，并执行其中的 `check()` 函数。

`use url` 预编译头指定的外部文件名在当前程序中必须唯一，用户不得指定不同 URL 地址的同名外部文件，否则在调用外部文件时就会发生混乱。

另外，`use url` 预编译头中的 URL 地址也可以是相对 URL 地址。相对 URL 的起始位置是当前程序文件所在的位置，并在此基础上根据 URL 进行定位。

如果 URL 地址中的字符包含有转义字符(即特殊字符的转义序列表示，请参见表 7.1)，

则 WMLScript 将根据转义要求进行转义。不过，程序在编译的时候编译器并不会对它们进行转义，而是在程序执行时完成，检查 URL 格式和 URL 地址的有效性。

8.3.2 访问权限

我们可以使用访问权限预编译头来保护文件的内容，实现访问控制。WMLScript 编程中，必须在调用外部函数之前使用访问权限预编译头声明外部文件的访问权限。不过，WMLScript 访问权限检查的缺省值是不进行检查，即 disabled。但访问权限一经声明，以后当调用外部函数的时候，编译器就会检查外部文件的访问权限，以决定调用者是否有权使用该文件及其内含函数。

访问权限预编译头的声明方法是 use access，其语法格式如下：

```
use access domain 操作域名 path 操作路径;
```

访问权限预编译头通过指定 domain 和 path 属性来决定编译器将要进行什么样的检查工作。如果文件有 domain 或者 path 属性，那么文件所在的 URL 就必须和属性中的值一致。比较时，域和路径都依据 URL 大写规则进行比较。具体的比较原则如下：

- (1) 操作域与 URL 中的域后缀相匹配。域后缀匹配是指所有的子域从后向前都必须一致。例如：www.wapforum.org 和 wapforum.org 相匹配，而与 forum.org 并不匹配。
- (2) 操作路径和 URL 中的路径前缀相匹配。路径前缀匹配是指从前向后都必须一致。例如：路径“/X/Y”与“/X”相匹配，而不是和“/XZ”相匹配。
- (3) 缺省的 domain 属性为当前的文件域，也就是“/”。

不过，为了简化编程，有时 WMLScript 并不需要知道外部文件的绝对路径，我们只需提供文件的相对 URL 即可，用户浏览器执行程序时可以把相对路径自动转换为绝对路径，并根据路径属性进行匹配。例如：如果访问权限预编译头及其指定属性为：

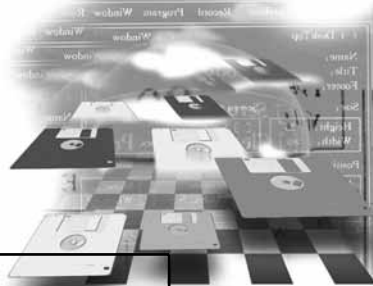
```
use access domain "wapforum.org" path "/finance";
```

则可以使用以下的路径来调用指定文件中的外部函数，它们都符合相对 URL 地址匹配规则：

```
http://wapforum.org/finance/money.cgi
```

```
http://www.wapforum.org/finance/markets.cgi
```


```
http://www.wapforum.org/finance/demos/packages.cgi?x+123&y+456
```



而以下的路径调用则非法的，因为它们或者操作域不对，或者 URL 地址不能与指定的相对 URL 相匹配：

```
http://www.test.net/finance
```

```
http://www.wapforum.org/internal/foo.wml
```

 需要强调指出的是，WMLScript 规定，同一程序中只能定义一个访问权限预编译头，否则就会导致编译错误。

8.3.3 Meta 信息

我们还可以通过预编译头的形式声明 WMLScript 文件的 Meta 信息。Meta 信息主要用于指定文件所需 Meta 属性的属性名(Property name)、属性值(Content)以及文件的配置(Scheme)信息，属性值都属于字符串类型的数据。Meta 信息预编译头使用 use meta 声明，其语法格式为：

```
usr meta 属性 该属性 Meta 信息；
```

Meta 的属性主要包括 Name、HTTP Equiv 和 User Agent 三种，下面我们分别讲解它们的声明方法。

(1) Name。该属性用于指定服务器使用的 Meta 信息。这些信息仅供服务器使用，用户浏览器并不理会这些信息。

例如，以下 Name 属性的 Meta 信息指定了服务器的创建时间：

```
use meta name "Created" "26-June-2000";
```

该信息只会作用于服务器，而不会影响用户浏览器的操作。

(2) HTTP Equiv。该属性用于指定需要解释为 HTTP 头的 Meta 信息。对于已经编译的文件来说，当它到达用户浏览器前，WMLScript 将根据 HTTP Equiv 属性指定的 Meta 信息将文件转换为 WSP 或 HTTP 的响应头，进行文件的解释和执行。

例如，以下声明的 http equiv 属性指定按照脚本语言的关键字来解释当前文件：

```
use meta http equiv "Keywords" "Script, Language";
```

(3) User Agent。该属性用于定义用户浏览器使用的数据类型。例如：

```
use meta user agent "Type" "Test";
```



它指定当前数据必须立即发送给用户浏览器，然后马上清除掉。

8.4 运行错误检测和处理

由于 WMLScript 函数旨在为 WAP 用户提供各种无线服务，所以它希望用户的浏览器在各种情况下都能够正常工作，因此有效的错误处理能力对它而言是十分重要的。除非万不得已，我们一般情况下不能终止程序的执行，所以即便是 WMLScript 语言本身没有提供错误处理机制，我们也需要为 WMLScript 程序运行提供一些工具来处理发生的错误或检查可能导致错误的操作。下面我们就讨论 WMLScript 程序中可能出现的错误，并给出相应的检测和解决办法。

8.4.1 错误检测

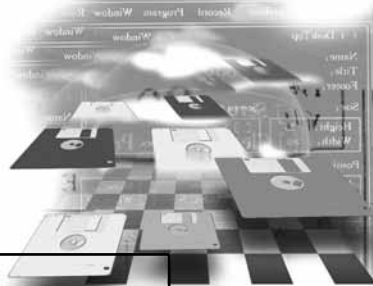
WMLScript 的错误检测工具允许我们在程序运行中检测可能出现的错误，不过错误检测工作可能会影响程序的正常运行。由于 WMLScript 是一种“弱类型”、即对数据类型要求不甚严格的语言，所以数据类型使用不当通常会引起错误。为此，WMLScript 专门提供了用于检测无效数据类型的工具，主要包括以下两类：

- (1) 检测给定变量的变量值为正确值的函数。这类函数均属于类型有效库函数，如 `Lang.isInt()`、`Lang.isFloat()`、`Lang.parseInt()` 和 `Lang.parseFloat()`。这些函数的具体用法我们将在下一章专门讲述。
- (2) 检测给定变量的变量值类型为正确类型的操作符。即 `typeof` 操作符和 `isvalid` 操作符，具体内容我们前面已经讲过(参见 7.5.6 节)，我们这里不再重述。

8.4.2 错误处理的两类情况

由于 WAP 手机内存有限、外部信号不好或数据运算溢出、下溢等原因，WMLScript 程序执行中可能会发生一些不能被错误检测机制所检测、也无法预防的错误。这种错误发生后 WMLScript 就会尝试着进行错误处理。为了研究与操作上的方便，我们通常把这些错误分成如下两类：

- (1) 致命错误。主要指那些能够导致程序中止执行的错误。由于 WMLScript 的函数通



常是在用户端调用，所以致命错误发生时程序会给用户端发出信号，用户端的浏览器就会据此给用户显示出有关错误信息。

(2) 非致命错误。这类错误一般不会导致程序运行中止。错误发生时将向程序返回一个错误值，WMLScript 程序将立即寻找相应的处理办法。

有关致命错误和非致命错误的类型及处理方法，我们下面分列两个标题进行介绍。

8.4.3 致命错误及其处理

WMLScript 的致命错误主要包括字节码错误、程序异常中止错误、内存耗尽错误和外部异常错误四种情况。虽然说这类错误一旦发生我们无法补救，但了解这类错误的实质有助于我们在程序设计时尽量避免编写这类代码，尽量减少这类错误。

字节码错误

这类错误都与 WMLScript 的字节码解释器所执行的字节码和指令有关，如内含元素错误、指令非法、指令参数非法，甚至指令无法执行等。

(1) 确认失败。当字节码解释器调用的外部文件无法通过确认有效性检查时，通常会会发生确认失败的错误。例如，在下述调用外部文件函数的语句中，如果外部文件的预编译头设置、函数名或参数类型不正确，就都有可能产生确认失败错误：

```
var a = 3*OtherScript#doThis(param);
```

这种错误仅当字节码进行确认有效性检查时才能发现，但那时为时已晚，解释器只能中止程序执行，并给用户返回错误信息。

避免这种错误的方法就是程序设计时仔细弄清楚外部文件的位置、函数名称及参数使用方法，正确书写语句代码和定义有关变量，并确保它们有效，而且能通过有效性检查。

(2) 库函数致命错误。每次调用 WMLScript 的库函数时，如果调用代码不对或函数参数类型不对，都有可能产生库函数致命错误。这会致使库函数调用失败，从而程序中止执行。例如下面这句调用就有可能产生此类错误：

```
var a = String.format(param);
```

库函数致命错误发生前无法检测，所以我们程序设计时一定要搞清楚各种库函数的调用方法。

(3) 函数参数无效。如果调用函数时，所用参数类型、数量、参变量值等与函数定义的



不能完好匹配，那么就会导致函数参数无效的误差。这种误差也是无法事先检测的，因此程序设计时要正确设置调用函数的各种参数。

(4) 找不到外部函数。如果外部文件没有指定，或者外部文件虽然指定但其中却不含调用的函数，都会导致无法找到外部函数的误差。有关外部文件的预编译设置和外部函数的调用方法，我们上一节已经讲过，只要编程时能够正确地应用那些方法，找不到外部函数的误差还是可以避免的。

(5) 无法加载外部文件。如果网络服务器不存在，或网络服务器故障，或网络通信存在问题，都有可能导致无法加载外部文件的误差，即加载外部文件的指令不能返回结果。这主要是由网络和服务环境等外在因素引起的误差，程序设计人员通常无能为力，只有通信环境顺畅，才可杜绝此类误差。

(6) 访问受限。如果用户没有权限访问某一外部文件，则当强行访问该外部文件时就会导致访问受限误差，并致使程序执行中止。因此，程序设计中涉及外部文件调用时，一定要根据用户可能的访问权限调用相应的文件。

(7) 堆栈下溢。如果程序代码误差，则每当程序试着取出空堆栈时通常会发生堆栈下溢的误差。这种情况只有在 WMLScript 字节码解释器生成程序的坏代码时才产生，属于 WMLScript 解释系统的问题，只有建设好稳定的编译和解释环境，才能避免这类误差。

程序异常中止误差

当 WMLScript 的函数调用 `Lang.abort()` 库函数时，如果函数调用不匹配，则可能使字节码中止执行，产生程序异常中止误差。例如，下面的语句就有可能导致异常中止误差：

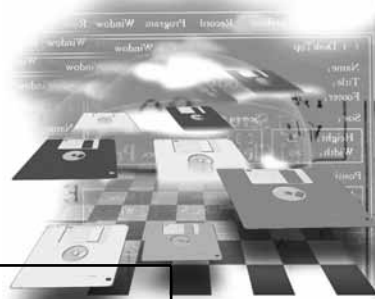
```
Lang.abort("Unrecoverable error");
```

避免这类误差的方法是熟悉并正确使用 `Lang.abort()` 库函数。有关该函数的用法我们下一章将专门介绍。

内存耗尽误差

内存耗尽误差与 WMLScript 解释器的动态操作行为和内存使用方法有关，可能导致堆栈溢出，也可能导致内存不足。

(1) 堆栈溢出。如果程序执行中试着向操作堆栈放入过多的变量，或者试着进行了多层的函数递归调用，那么都有可能导致堆栈溢出。例如，下面这种函数的递归调用就可能会导致堆栈溢出误差，因为它使得变量 `x` 的值不断地增加 1，直到 `x` 值大到用尽内存为止：



```
function f(x) { f(x+1); };
```

避免这种错误的方法就是在程序设计时，尽量减少变量数量和递归调用层数，并且杜绝死循环的程序。对于函数的递归调用，我们的建议是最好不采用函数的递归调用，而利用函数嵌套等方法解决。

(2) 内存不足。WMLScript 程序执行过程中，如果操作系统不能为解释器分配充足的内存，供程序解释和执行之用，那么就会导致内存不足的致命错误。

例如，下面的函数就可能会导致内存不足：

```
function f(x) {  
    x=x+"abcdefghijklmnopqrstuvxyz";  
    f(x);  
};
```

该程序执行了一个死循环，不断地把“abcdefghijklmnopqrstuvxyz”这个字符串追加到字符串变量 x 中，直到 x 的值超过系统内存承受能力而致死机时为止。

一般地，编程中应尽量减小单个数据的规模和内存资源的占用，并避免死循环程序，这样通常能够有效地防止内存不足错误的发生。

外部异常错误

外部异常错误大都是由 WMLScript 字节码编译器的外部因素引起的，比如用户或系统等因素。

(1) 用户引起的外部异常错误。如果在程序运行中，用户要中止程序的执行并发出了中止指令，比如按了 Reset 键，那么此时程序就会中止执行，并给用户返回相应的错误信息。

(2) 系统引起的外部异常错误。主要是指由于系统供电不足、断电或设备损坏等因素造成的异常中止程序执行的错误。避免这类错误也没有什么好方法，只有依赖于程序的外部系统环境稳定、可靠，减少意外事件的发生。

8.4.4 非致命错误及其处理

非致命错误主要包括计算错误、常量错误和转换错误等，这些错误不会导致程序崩溃，但会返回一个 invalid 的无效值或 0.0，说明错误性质。

计算错误

计算错误都与 WMLScript 所支持的数学运算功能有关，比如除数为零、整数溢出、浮点数溢出、浮点数下溢等。

(1) 除数为零。当在程序中出现 0 除整数、浮点数或其他数据时，都会导致计算错误。

例如下面的简单程序就会出现除数为零的计算错误：

```
var a = 10;  
var b = 0;  
var x = a / b;  
var y = a div b;  
var z = a % b;  
a /= b;
```

避免这种错误的方法并不复杂，只需在程序设计时仔细分析变量数据，尤其是作为除数的变量的值，并通过 if 条件语句等另外处理这种情况即可。

(2) 整数溢出。程序中执行整数操作时有可能出现整数计算结果溢出的错误。这种错误通常是整数经计算后产生的，所以编程阶段一般难以发现。不过在程序调试阶段，我们可以通过不断测试比较大的整数运算，来检测是否会出现整数溢出的错误。如果出现，就要对程序中的整数运算做适当的调整，以避免整数无限增大的情况。

例如，下面对 WMLScript 所能接受的最大整数的求和运算就会导致整数溢出的错误：

```
var a = Lang.maxInt();  
var b = Lang.maxInt();  
var c = a + b;
```

(3) 浮点数溢出。与整数溢出一样，程序中执行浮点数操作时有可能出现浮点数计算结果溢出的错误。例如，下面对两个极大浮点数的求积运算就会导致浮点数溢出的错误：

```
var a = 1.6e308;           // 1.6e308 = 1.6 × 10308，这是非常大的数！  
var b = 1.6e308;  
var c = a * b;
```

这种错误一般也是在浮点数计算后产生的，编程阶段很难发现。所以在程序调试阶段，



我们需要通过不断测试较大浮点数的运算，来检测是否会出现此类溢出错误。如是，则调整程序，避免这类溢出错误。

(4) 浮点数下溢。当程序中对非常小的浮点数进行操作时，就有可能导致浮点数下溢的错误。这种错误通常也是经计算后产生的，所以编程阶段一般很难发现。发生这种错误时的返回值为 0.0。

例如，下面对 WMLScript 所能接受的最精确(也就是最小)浮点数的求积运算就会导致浮点数下溢的错误：

```
var a = Float.precision();  
var b = Float.precision();  
var c = a * b;
```

为了避免这种错误，我们必须在编程阶段仔细分析浮点运算结果的各种可能性，尽量绕开下溢的情况，或者针对返回的 0.0 结果给出相应的处理。

常量错误

常量错误主要是由浮点常数非数值性、浮点常数无穷及浮点变量非法等原因引起的，通常与程序运行中的变量赋值、常量运算等操作有关。避免此类错误的主要方法是分析程序中的数据处理代码，了解数据的变化趋势，并给出相应的转移方法或规避方法。

(1) 浮点常数并非数值。如果一个常量类型为浮点型，但程序运行中却赋给它一个非数值型的数据并参与和浮点数相关的运算，或具有非数值型数据的常量强行进行向浮点型数的转换，那么都会引起此类错误。出现错误时，程序将返回 invalid 值作为错误信息。

(2) 浮点常数无穷。如果程序运行中生成了正无穷或负无穷的浮点数，则当对这些浮点数进行操作时就会产生浮点常数无穷的误差。它的返回值也为 invalid。

(3) 浮点变量非法。如果程序试图使用浮点数时，赋值变量或所处程序环境仅要求数据为整型，则会导致浮点变量非法的错误。它的返回值也为 invalid。

转换错误

转换错误是与 WMLScript 所支持的数据类型自动转换有关，如整数太大、浮点数太大或太小等错误。

(1) 整数太大。其他数据类型向整数转换时，如果可能生成的绝对值整数太大(含正整数、负整数)，则会导致整数太大的转换错误。它的返回值也为 invalid。下述语句就是一个

[illegible]

(2) 浮点数太大。与整数太大的错误一样，其他数据类型向浮点数转换时，如果可能生绝对值浮点数太大(含正、负浮点数)，则会导致浮点数太大的转换错误。例如：

```
var a = -"99999999.999999999999e999999":
```

(3) 浮点数太小。当其他数据类型向浮点数转换时，如果可能生成的绝对值浮点数太小(含正、负浮点数)，则会导致浮点数太小的转换错误。例如：

```
var a = -"0.01e-99";
```

本章小结

本章我们讲解了 WMLScript 的语句功能和使用方法，讲解了函数分类及调用方法，介绍了 WMLScript 的预编译处理、错误检测手段和具体的解决办法。这些内容是 WMLScript 脚本程序设计的关键内容，尤其是语句使用方法和函数调用方法，希望大家能够认真掌握，并能熟练地编写 WMLScript 脚本程序。



第9章 WMLScript 库及库函数

WMLScript 语言提供了强大的函数功能，利用这些函数我们能够快速实现许多重要的操作及处理，所以 WMLScript 编程中使用函数是我们提高开发效率的有力手段。WMLScript 的函数均隶属于相应的库，因此这些函数也称为库函数。WMLScript 有标准库函数，也有非标准库函数；标准库函数主要包含在 Lang、Float、String、URL、WMLBrowser 和 Dialogs 几种库中，非标准库函数主要包含在 Debug 等库中。本章我们就详细讲解这些库的组成及库函数的功能和使用方法，并在最后分析几个使用 WML、WMLScript 开发的 WAP 应用实例。

9.1 Lang 库及其函数

WMLScript 的 Lang 库包含了一些与 WMLScript 语言核心数据相关的一些函数，使用这些函数可以实现 WMLScript 程序中的数据处理，如 abort、abs、characterSet、exit、float、isFloat、isInt、max、maxInt、min、minInt、parseFloat、parseInt、random 及 seed 等。

9.1.1 abs 函数

abs 函数用于返回给定数值的绝对值。如果给定的数值是个整数，那么返回值就是个整数；如果给定的数值是浮点数，那么返回的数值就是浮点数。如果给定的数值不符合 WMLScript 的数据类型要求，则返回 invalid。abs 函数的语法格式为：

Lang.abs(value)

它只有一个参数 value，可取值为数值。

例如，下面两行语句就是使用 abs 函数的简单例子：

```
var a = -6;  
var b = Lang.abs(a);           //b = 6
```

第 1 条语句将-6 赋给变量 a，第 2 条语句利用 abs 函数求 a 的绝对值并把结果赋给变量 b，所以此时 b 的结果为+6。

9.1.2 min 函数

min 函数用于求出两个给定数值中的较小数值。求出该数即选中该数并返回给调用 min 函数的语句或函数，返回数的类型和值与选中数的类型及值完全一致。min 函数的语法格式如下：

```
Lang.min(value1, value2)
```

它有两个参数 value1 和 value2，它们的取值必须是数值。如果它们取值的数据类型不同，则 min 函数将根据 WMLScript 的整数或浮点数的自动转换规则，将它们的值转换为同一种类型的数据，然后再进行比较，并从中选取较小的那个数。如果两个数相等，则选取第 1 个参数的数值。

如果 value1 和 value2 的数值类型不能转换为可以比较的数据类型，则 min 函数就会放弃它们的比较，而返回 invalid，表示参数无效，不能比较。

例如，下面我们给出了几个使用 min 函数进行比较的语句，我们同时注释了它们的比较结果，大家可以从中分析一下 min 函数的工作方式：

```
var a=-3;  
var b=Lang.abs(a);  
var c=Lang.min(a,b);           // 比较结果是 c=-3  
var d=Lang.min(45, 76.3);      // 比较结果是 d=45 (整数)  
var e=Lang.min(45, 45.0);      // 比较结果是 e=45 (整数，因为要选择第 1 个数)  
var e=Lang.min(45.0, 45);      // 比较结果是 e=45.0 (浮点数，因为要选择第 1 个数)
```



9.1.3 max 函数

与 min 函数相反，max 函数用于求出两个给定数值中的较大数值。求出的数即是选中的较大的数，该数将返回给调用 max 函数的语句或函数，返回数的类型和值与选中数的类型和值完全一致。max 函数的语法格式如下：

```
Lang.max(value1, value2)
```

它有也两个参数 value1 和 value2，它们的取值必须是数值。如果它们取值的数据类型不同，则 max 函数将根据 WMLScript 的整数或浮点数的自动转换规则，将它们的值转换为同一种类型的数据，然后再进行比较，并从中选取较大的那个数。如果两个数相等，则选取第 1 个参数的数值。

如果 value1 和 value2 的数值类型不能转换为可以比较的数据类型，则 max 函数就会放弃它们的比较，而返回表示参数无效、不能比较的无效值 invalid。

作为举例，我们给出几个使用 max 函数进行比较的语句，供大家参考：

```
var a=-3;
var b=Lang.abs(a);
var c=Lang.max(a,b);           // 比较结果是 c=3
var d=Lang.max(45.5, 76);      // 比较结果是 d=76 (整数)
var e=Lang.max(45.0, 45);      // 比较结果是 e=45.0 (浮点数，因为是取第 1 个数)
var e=Lang.max(45, 45.0);      // 比较结果是 e=45 (整数，因为是取第 1 个数)
```

9.1.4 parseInt 函数

parseInt 函数用于返回字符串所定义的整数的数值。其前提是字符串中的内容及格式必须符合 WMLScript 关于字符串向整数转换的规则(参见 7.6 节)，并且字符串分析时只要第 1 个字符不是正号(+)、负号(-)或十进位数字，那么转换分析就会结束。如果字符串无法转换成整数，则 parseInt 函数将返回无效值 invalid。parseInt 函数的语法格式为：

```
Lang.parseInt(value)
```

它只有一个参数 value，其取值只能是字符串。下述两条语句给出了转换的例子：

```
var i=Lang.parseInt("1234");           // 转换后 i=1234
var j=Lang.parseInt("100 m/s");        // 转换后 j=100
var k=Lang.parseInt("Abt25Z6vd");      // 无法转换，返回 k=invalid
```

9.1.5 parseFloat 函数

parseFloat 函数用于返回字符串所定义的浮点数的数值。与 parseInt 函数类似，parseFloat 函数成功转换的其前提是，字符串中的内容及格式必须符合 WMLScript 关于字符串向浮点数转换的规则(参见 7.6 节)。如果字符串无法转换成浮点数或者程序环境不支持浮点数，则 parseFloat 函数将返回无效值 invalid。parseFloat 函数的语法格式为：

```
Lang.parseFloat(value)
```

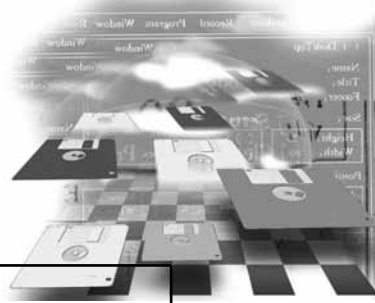
它只有一个参数 value，其取值只能是字符串。下述几条语句给出了字符串向浮点数转换的例子：

```
var a=Lang.parseFloat("123.7");        // 转换结果为 a=123.7
var b=Lang.parseFloat("+7.34e2 Hz");    // 转换结果为 b=7.34e2
var c=Lang.parseFloat("70e-2 F");       // 转换结果为 c=70.0e-2
var e=Lang.parseFloat(" 100 ");         // 转换结果为 e=100.0
var f=Lang.parseFloat("Number:5.5");    // 无法转换，结果为 f=invalid
var g=Lang.parseFloat("7.3e meters");   // 无法转换，结果为 g=invalid
var h=Lang.parseFloat("7.3e- m/s");      // 无法转换，结果为 h=invalid
```

9.1.6 isInt 函数

如果给定的值 value 能够被 parseInt 函数转换为整数类型，则使用 isInt 函数对该值 value 测试时将返回 true，即说明 value 能够转化为整数；否则，其他情况一律返回 false 或 invalid。isInt 函数的语法为：

```
Lang.isInt(value)
```

它只有一个参数 `value`，可以取任何类型的数据值。它的返回值是布尔型的 `true|false` 或 `invalid`。

作为举例，我们下面给出几条语句说明 `isInt` 函数的用法：

```
var a=Lang.isInt("-123");           // 返回结果为 true
var b=Lang.isInt("123.33");         // 返回结果为 true
var c=Lang.isInt("string");         // 返回结果为 false
var d=Lang.isInt("#123");           // 返回结果为 false
var e=Lang.isInt(invalid);          // 返回结果为 invalid
```

9.1.7 isFloat 函数

如果给定的值 `value` 能够被 `parseFloat` 函数格式化为浮点数类型，则使用 `isFloat` 函数对该值 `value` 测试时将返回 `true`，即说明 `value` 能够转化为浮点数；否则，其他情况一律返回 `false` 或 `invalid`。`isFloat` 函数的语法为：

```
Lang.isFloat(value)
```

它只有一个参数 `value`，可以取任何类型的数据值。它的返回值是布尔型的 `true|false` 或 `invalid`。其中，返回无效值 `invalid` 的条件是，当前系统不支持浮点操作。

例如，下面几条语句可以对给定的参数值进行测试，并判断出能否转换为浮点数：

```
var a=Lang.isFloat("-123");          // 返回值为 true
var b=Lang.isFloat("123.33");        // 返回值为 true
var c=Lang.isFloat("string");         // 返回值为 false
var d=Lang.isFloat("#123.33");        // 返回值为 false
var e=Lang.isFloat(invalid);          // 返回值为 invalid
```

9.1.8 maxInt 函数

该函数用于返回最大的整数值。由于 WMLScript 所能支持的最大整数值为 2147483647，所以 `maxInt` 函数的返回值即为 2147483647。`maxInt` 函数的语法格式为：

```
Lang.maxInt()
```



它没有参数，返回值是固定的，即 2147483647。比如，下述语句即可返回 WMLScript 支持的最大整数值：

```
var a=Lang.maxInt( );
```

9.1.9 minInt 函数

该函数用于返回最小的整数值。由于 WMLScript 所能支持的最小整数值为 -2147483647，所以 minInt 函数的返回值即为 -2147483647。minInt 函数的语法格式为：

```
Lang.minInt( )
```

它没有参数，返回值也是固定的，即 -2147483647。比如，下述语句即可返回 WMLScript 支持的最小整数值：

```
var a=Lang.minInt( );
```

9.1.10 float 函数

如果当前的程序环境支持浮点型数据的处理，则使用 float 函数判断后可以返回 true，否则返回 false。比如含有合法浮点数的字符串可以向浮点数转换，说明当前程序环境支持浮点数处理，则使用 float 函数测试时将返回 true。float 函数的语法为：

```
Lang.float( )
```

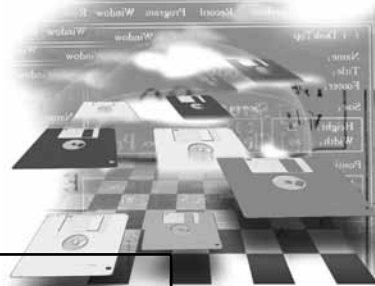
它没有参数，返回值类型为布尔型。例如，使用下述语句测试当前环境是否支持浮点数：

```
var floatsSupported = Lang.float( );
```

9.1.11 exit 函数

exit 函数用于终止 WMLScript 字节码解释器的解释执行，并将控制权转移到原调用程序的解释器上。该函数本身没有返回值，但它可以返回一个用户指定的值 value。exit 函数的语法格式为：

```
Lang.exit(value)
```



其中参数 value 由用户指定，可以是任何类型的数据。例如，下面的语句都可终止解释器的执行，并返回给定的 value 值：

```
Lang.exit("Value: " + myVal);      // 返回值是一个字符串
Lang.exit(invalid);                // 返回值为 invalid
```

一般来说，exit 函数仅当程序需要中断的时候才可以使用，正常地中断 WMLScript 的解释器。否则，要尽量少用这种终止程序执行的函数。

9.1.12 abort 函数

该函数可使程序退出当前的 WMLScript 字节码解释器，返回到调用当前程序的 WMLScript 解释器，同时返回错误描述(errorDescription)。尤其当 WMLScript 函数或外部文件的调用遇到非常严重的错误时，abort 函数可以用来执行异常退出。如果返回的错误描述是 invalid，该函数还同时自动转换该值为"invalid"字符串。

abort 函数的语法格式为：

```
Lang.abort(errordescription)
```

它只有一个参数，即 errordescription，用于表示错误描述信息，数据类型为字符串型。例如，下述语句便利用了 abort 函数的退出解释器功能：

```
Lang.abort("Error: " + errVal);    // 其中 Error 的值是个字符串
```

9.1.13 random 函数

random 函数可以根据给定参数 value 的值生成并返回一个正的随机数。该随机数是一个大于等于 0 且小于等于 value 值的整数。random 函数的语法格式为：

```
Lang.random(value)
```

它只有一个参数 value，取值数据类型必须为整型。如果用户给出的是浮点数，则 random 函数首先将它转换为整型，然后再生成随机数。如果 value 的取值为 0，则 random 函数将返回 0；如果 value 的取值小于 0，则 random 函数将返回无效值 invalid。

```
var a=10;
var b=Lang.random(5.1)*a;           // 返回结果为 b=0..50(这是随机的，用户测试时未必与之相同)
var c=Lang.random("string");        // 返回结果为 c=invalid
```

9.1.14 seed 函数

当给定参数 value 的值为 0 或者是正整数时,seed 函数将根据该参数的值来初始化伪随机数序列(pseudo-random number sequence)并返回一个空字符串。伪随机数序列就是我们通常所称的随机数种子, WMLScript 的 random 函数将根据该随机数种子来求随机数。seed 函数的语法格式为:

```
Lang.seed(value)
```

它只有一个参数 value,其取值需为 0 或正整数。如果 value 的值为浮点数,则 seed 函数将首先将它转换为整数,然后再进行伪随机数序列的初始化。如果给定参数 value 的值不为 0,或者不是正整数,或者不能转换为正整数,那么 seed 函数就不能根据 value 的值初始化伪随机数序列,即此时不能改变 random 函数所使用的随机数种子。

下面几条语句给出了使用 seed 函数的例子:

```
var a=Lang.seed(123);               // 结果是 a=""
var b=Lang.random(20);              // 结果是 b=0..20
var c=Lang.seed("seed");            // 结果是 c=invalid (所以此时不能改变随机数种子)
```

9.1.15 characterSet 函数

该函数用于返回 WMLScript 解释器所能支持的字符集,返回值是一个整数值,代表 IANA(Internet Assigned Numbers Authority)的一种标准字符集。characterSet 函数的语法格式如下:

```
Lang.characterSet()
```

例如,下述语句就可以返回当前 WMLScript 解释器支持的字符集:

```
var charset = Lang.characterSet();
```

返回的结果是 charset = 4, 4 代表 latin1 字符集。



9.2 Float 库及其函数

WMLScript 的 Float 库提供了一系列典型的用于浮点数运算和处理的功能函数。使用 Float 库函数的前提条件是 WMLScript 程序环境及设备能够支持浮点数运算,如果不支持浮点数的运算,那么 Float 库函数所有的返回值都将是无效值 invalid。Float 库函数主要包括 ceil、floor、int、maxFloat、minFloat、pow、round 与 sqrt,本节就详细介绍它们的功能和用法。

9.2.1 int 函数

int 函数用于求出并返回给定参数 value 值中的整数部分。如果 value 是个整数,那么返回值就是 value 本身;如果 value 是浮点数,则只返回 value 中的整数部分。如果 value 无法转换为整数或无法求出整数部分,则 int 函数将返回无效值 invalid。int 函数的语法格式为:

```
Float.int(value)
```

下述几条语句给出了使用 int 函数的例子:

```
var a = 3.14;  
var b = Float.int(a);           // 返回结果为 b = 3  
var c = Float.int(-2.8);        // 返回结果为 c = -2  
var d = Float.int(156.128);      // 返回结果为 d = 156
```

9.2.2 floor 函数

floor 函数用于求出并返回所有小于参数 value 值中的最大整数。如果 value 已经是个整数,那么返回的结果就是 value 本身。floor 函数的语法格式为:

```
Float.floor(value)
```

它的参数 value 的值必须是个有效的数,否则,floor 函数只能返回无效值 invalid。使用 floor 函数的例子见下:

```
var a = 3.14;  
var b = Float.floor(a);         // 返回结果为 b = 3
```



```
var c = Float.floor(-2.8);      // 返回结果为 c = -3  
var d = Float.floor(156.128);  // 返回结果为 d = 156
```

9.2.3 ceil 函数

ceil 函数用于求出并返回所有不小于给定参数 value 值中的最小整数。如果 value 是整数，那么返回的结果就是 value 本身。ceil 函数的语法格式为：

```
Float.ceil(value)
```

它的参数 value 的值也必须是个有效的数，否则 ceil 函数就会返回无效值 invalid。

作为举例，我们给出几个使用 ceil 函数的例子：

```
var a = 3.14;  
var b = Float.ceil(a);          // 返回结果为 b = 4  
var c = Float.ceil(-2.8);       // 返回结果为 c = -2  
var d = Float.ceil(156.128);    // 返回结果为 d = 157
```

9.2.4 pow 函数

pow 函数有 value1 和 value2 两个参数，它的功能就是计算并返回以参数 value1 的值为底，以参数 value2 的值为指数的幂。如果 value1 的值是负数，那么要求 value2 的值必须是整数。pow 函数的语法格式如下：

```
Float.pow(value1, value2)
```

它的两个参数 value1 和 value2 的值必须是数值型的数，否则，pow 函数就无法计算它们的幂，此时就只能返回无效值 invalid。

另外，如果 value1 的值为 0 并且 value2 的值小于 0，或者 value1 的值小于 0 而 value2 的值不是整数，则 pow 函数无法计算它们的幂，并会返回无效值 invalid。

pow 函数的使用举例如下：

```
var a = 3;  
var b = Float.pow(a, 2);        // 计算结果为 b = 9
```



```
var c = 2.78;
var d = Float.pow(c, 3);           // 计算结果为 d = 21.484952
var e = Float.pow(-2, 3);          // 计算结果为 e = -8
var f = Float.pow(0, -3);           // 计算结果为 f = invalid
var g = Float.pow(-2, 3.5);         // 计算结果为 g = invalid
```

9.2.5 round 函数

round 函数用于求出并返回最接近给定参数 value 值的整数值。如果两个整数都和 value 值等距离，那么 round 函数将返回较大的那个整数值。如果 value 是一个整数，则返回的就是 value 本身。round 函数的语法格式为：

```
Float.round(value)
```

如果参数 value 的值不是有效的数，那么 round 函数就会返回无效值 invalid。

使用 round 函数的例子如下：

```
var a = Float.round(3.5);           // 计算结果为 a = 4；4 和 3 与 3.5 等距离，取较大的数 4。
var b = Float.round(-3.5);          // 计算结果为 b = -3
var c = Float.round(0.5);           // 计算结果为 c = 1；1 和 0 与 0.5 等距离，取较大的数 1。
var d = Float.round(-0.5);          // 计算结果为 d = 0
```

9.2.6 sqrt 函数

sqrt 函数用于计算并返回给定参数 value 值的近似平方根。其语法格式如下：

```
Float.sqrt(value)
```

它的参数 value 取值需为浮点数，返回值也为浮点数。不过，如果 value 的值无法求出平方根，比如 value 的值是个负数，则 sqrt 函数将返回无效值 invalid。

sqrt 函数的计算例子如下：

```
var a = 4;
var b = Float.sqrt(a);              // 计算结果为 b = 2.0
```

```
var c = Float.sqrt(5);           // 计算结果为 c = 2.2360679775
```

9.2.7 maxFloat 函数

maxFloat 函数用于返回单精度浮点数所支持的最大浮点数。其语法格式为：

```
Float.maxFloat( )
```

它没有参数。由于单精度浮点数所支持的最大浮点数为 3.40282347E+38，所以该函数的返回值即为浮点数 3.40282347E+38。

例如，下述语句即可使用 maxFloat 函数返回最大的单精度浮点数：

```
var a = Float.maxFloat( );
```

9.2.8 minFloat 函数

minFloat 函数用于返回单精度浮点数所支持的最小浮点数。其语法格式为：

```
Float.minFloat( )
```

该函数也没有参数。由于单精度浮点数所支持的最小浮点数为 1.17549435E-38，所以该函数的返回值即为浮点数 1.17549435E-38。

比如，下述使用 minFloat 函数的语句即可求出最小的单精度浮点数：

```
var a = Float.minFloat( );
```

9.3 String 库及其函数

WMLScript 的 String 库包含了一系列的用于字符串处理的函数。前文述及，字符串可以看作是一个字符数组。在该数组中，每个字符都有一个下标序号，其中左起第 1 个字符的序号是 0，其他字符的序号由左向右依次递增。使用 String 库函数处理字符串时，经常需要利用字符的序号实现对字符的操作。字符串也是有长度，其长度就是数组中字符的个数。

另外，为了便于 String 库函数处理字符串，WMLScript 允许我们可以使用指定的分隔符来分割字符串的内容。这样，某些 String 库函数就可以通过指定分隔符和字符序号进行



操作。WMLScript 规定，转义字符不能用作分割符。

考虑到有些特殊字符，比如回车符等，是不能直接放在字符串中的，因此 WMLScript 规定了一些“空字符”，利用这些空字符就可把相应的特殊字符放在字符串中进行处理。WMLScript 规定的空字符主要有 6 种，如表 9.1 所示。

表9.1 WMLScript的空字符

空字符	表示的特殊字符
TAB	水平制表符
VT	垂直制表符
FF	打印机进纸
SP	空格
LF	换行
CR	回车

String 库包含的函数比较多，如 charAt、compare、elementAt、elements、find、format、insertAt、isEmpty、length、removeAt、replace、replaceAt、squeeze、subString、toString、trim 等，下面我们就分而述之。

9.3.1 length 函数

length 函数用于计算并返回给定字符串的长度，即字符串所含字符的个数。其语法格式为：

```
String.length(string)
```

它的参数 string 取值只能为合法的字符串或者能够转换为字符串的其他类型有效数据，这样计算后可以返回表示字符串长度的整数值，否则就会返回无效值 invalid。

下面几条语句给出了使用 length 函数的例子：

```
var a = "ABC";  
var b = String.length(a);           // 返回结果为 b = 3  
var c = String.length("");          // 返回结果为 c = 0  
var d = String.length(342);         // 返回结果为 d = 3
```

9.3.2 isEmpty 函数

该函数用于判断给定的字符串是否为空。如果字符串为空，那么就返回 true，否则返回 false。如果给定的参数值并不是合法的字符串，则返回无效值 invalid。isEmpty 函数的语法格式为：

```
String.isEmpty(string)
```

作为举例，我们给出判定字符串是否为空串的几个例子：

```
var a = "Hello";  
var b = "";  
var c = String.isEmpty(a);           // a 不是空串，所以 c = false  
var d = String.isEmpty(b);           // b 是空串，所以 d = true  
var e = String.isEmpty(true);        // true 可以转换为字符串且不为空，所以 e = false
```

9.3.3 charAt 函数

charAt 函数的功能是利用给定的参数 index 从给定的字符串 string 中返回一个长度为 1 的新字符串，该字符串即是 string 在序号 index 处所包含的字符。如果 index 是浮点数，则 WMLScript 首先利用 Float.int() 函数将它转换为整数，然后再确定新字符串。charAt 函数的语法格式如下：

```
String.charAt(string, index)
```

其中的参数 string 需取字符串型的数据，它是被操作的字符串；参数 index 需取数值型的数据，它是要取的字符在 string 中的序号。如果 charAt 函数的参数不满足这些要求，那么它的返回值就会是无效值 invalid。而且，如果 index 超出字符串 string 的范围，即 index 比 string 的长度值还要大，那么 charAt 函数就会返回一个空字符串("")。

```
var a = "Monday, May 24";  
var b = String.charAt(a, 0);          // 序号为 0 的字符是串中第 1 个字符，故 b = "M"  
var c = String.charAt(a, 1);          // 序号为 1 的字符是串中第 2 个字符，故 c = "o"
```

```

var d = String.charAt(a, 2);           // 序号为 2 的字符是串中第 3 个字符，故 d = "n"
var e = String.charAt(a, 5);           // 序号为 5 的字符是串中第 6 个字符，故 e = "y"
var f = String.charAt(a, 100);          // 100 超出字符串的范围，故 f = ""
var g = String.charAt(34, 0);           // 34 可转换为字符串再求，故 g = "3"
var h = String.charAt(a, "first");      // 不符合参数类型规定，所以 h = invalid

```

9.3.4 subString 函数

subString 函数的功能是根据给定的开始序号参数 `startIndex` 和欲取字符串的长度参数 `length`，从给定的字符串 `string` 中提取生成一个子字符串。如果 `startIndex` 比 0 还小，那么 subString 函数就从序号 0 处提取子串。如果 `length` 的值比从序号 `startIndex` 开始所剩余的字符个数还要多，那么 subString 函数就会把剩余的字符作为子串返回。subString 函数的语法格式为：

```
string.subString(string, startIndex, length)
```

它有 3 个参数，各参数取值及含义如下：

- (1) `string` 参数的取值范围限于字符串或可以合法转换为字符串的数据，它是用于提取子串的父亲串；
- (2) `startIndex` 参数的取值范围为数值，它是提取子串在父串中开始的序号，即从该序号指定字符开始，提取作为子串的字符；
- (3) `length` 参数的取值也需要为数值，它是提取子字符串的长度。

如果 `startIndex` 或者 `length` 是浮点数，那么 WMLScript 将首先利用 `Float.Int()` 函数将它们转换为整数，然后再通过 subString 函数进行子字符串的提取操作。

如果各参数取值均能符合要求并有意义，则 subString 函数将返回提取的子字符串，否则就会返回无效值 `invalid`。

另外，如果 `startIndex` 超过原字符串的长度，那么 subString 函数将返回一个空字符串。如果 `length` 小于 0，则也会返回一个空字符串。

使用 subString 函数的例子如下：

```

var a = "ABCD";
var b = String.subString(a, 1, 2);      // 从序号为 1 的第 2 个字符开始取，取 2 个字符，故 b = "BC"

```

```
var c = String.subString(a, 2, 5);    // 从序号为 2 的第 3 个字符开始取, 取 5 个字符, 由于 5 超过
                                     // 剩余字符个数, 所以取剩余的字符, 故 c = "CD"
var d = String.subString(1234, 0, 2); // 转换 1234 为字符串, 并从第 1 个字符开始取 2 个, 故 d = "12"
```

9.3.5 find 函数

find 函数用于从给定的字符串中寻找并返回第一处匹配给定子字符串开始的序号。如果字符串不能包含与该子字符串相匹配的部分, 则返回-1。两个字符串相匹配就是指两个字符串是完全一样的, 包括大小写一样。例如, 字符串"abcd"与"abcd"就是互相匹配的, 字符串"myabcdef"中就包含有与"abcd"相匹配的子字符串。

find 函数的语法格式为:

```
String.find(string, subString)
```

它的两个参数 string 和 subString 的取值都是字符串或可以合法地转换为字符串的数据, 其中 string 是用于从中寻找子字符串的字符串, subString 是想要寻找的子字符串。

仍以字符串"myabcdef"和"abcd"为例来说, 由于字符串"myabcdef"中包含有与"abcd"相匹配的子字符串, 该子字符串在"myabcdef"中开始的序号是 2, 即从第 3 个字符开始, 所以 String.find("myabcdef", "abcd")返回的结果为 2。

由于字符串中的字符序号都是整数, 所以 find 函数能够运算时返回的结果都是整数。如果给定的参数值不合法, find 函数无法寻找子字符串的序号, 则会返回无效值 invalid。

作为举例, 我们给出使用 find 函数的几个例子:

```
var a = "abcde";
var b = string.find(a, "cd");    // 变量 a 中含有 cd 且从第 3 个字符开始, 故 b = 2
var c = string.find(34.2, "de"); // 无法匹配, 所以 c = -1
var d = string.find(a, "qz");    // 无法匹配, 所以 d = -1
var e = string.find(34, "3");    // 34 转换为字符串后第 1 个字符与"3"匹配, 故 e = 0
```



9.3.6 replace 函数

replace 函数用于在一个给定的字符串中，使用一个新的子字符串替换其中所有旧的子字符串，而且替换是严格区分大小写的。replace 函数的语法格式如下：

```
String.replace(string, oldSubString, newSubString)
```

它有 3 个参数，各参数的取值都是字符串或可以合法转换为字符串的数据。string 参数代表给定的字符串，oldSubString 参数代表字符串 string 中含有的将被替换掉的旧字符串，newSubString 参数代表用于替换的新字符串。replace 函数的功能就是在 string 字符串中，将凡是有 oldSubString 的地方，都以 newSubString 来代替。

例如，下述几条语句就给出了使用 replace 函数替换字符串的例子：

```
var a = "Hello Christina. What is up Christina?";  
var newName = "Marie"  
var oldName = "Christina"  
var c = String.replace(a, oldName, newName);    // 将给定字符串中的 Christina 一律替换为 Marie ,  
        // 所以结果为 c = "Hello Marie. What is up Marie?"  
var d = String.replace(a, newName, oldName);    // 将给定字符串中的 Marie 一律替换为 Christina ,  
        // 但其中没有 Marie , 所以不替换, 结果为 d = "Hello Christina. What is up Christina?"
```

9.3.7 elements 函数

elements 函数用于计算并返回给定字符串 string 中被分隔符 separator 所分割得到的元素个数。WMLScript 规定空字符串也是有效的分割元素，而字符串至少是个空字符串，所以字符串至少含有 1 个分隔元素，因此 elements 函数的返回结果不会是一个小于或等于 0 的数。elements 函数的语法格式为：

```
String.elements(string, separator)
```

它的参数 string 和 separator 的取值均为字符串，separator 字符串中只有第 1 个字符才用作分割符。如果两个参数的取值均符合要求，则 elements 函数可返回一个代表分隔元素个数的整数值，否则就会返回无效值 invalid。特别是，如果 separator 是空字符串，那么 elements

函数也会返回 invalid。

下面我们给出使用 elements 函数的几个例子，同时给出了返回结果：

```
var a = "My name is Joe; Age 50;";
var b = String.elements(a, " ");           // 分隔符为空格，故 b = 6
var c = String.elements(a, ";");          // 分隔符为 “;”，故 c = 3
var d = String.elements("", ";");         // 分隔符为 “;” 而被分字符串为空串，故 d = 1
var e = String.elements("a", ";");        // 分隔符为 “;”，故 e = 1
var f = String.elements(";", ";");        // 分隔符为 “;”，故 f = 2
var g = String.elements(";;;", ";");      // 分隔符实取为 “;”，故 g = 4
```

9.3.8 elementAt 函数

elementAt 函数用于在给定的字符串中，搜索并返回由序号 index 指定的元素，该元素使用 separator 定义的分隔符来分割。elementAt 函数的语法格式如下：

```
String.elementAt(string, index, separator)
```

它有 3 个参数，其中 string 是给定的字符串，index 是元素的序号，separator 是元素的分割符。elementAt 函数的返回值是字符串或无效值 invalid。

elementAt 函数工作的一般规则是：

- (1) 如果 index 比 0 小，那么 elementAt 函数就会返回 string 中的第 1 个元素；
- (2) 如果 index 比元素的个数还多，那么就返回最后 1 个元素；
- (3) 如果 string 是空字符串，elementAt 函数返回的也将是 1 个空字符串；
- (4) 如果 index 是浮点数，那么 WMLScript 将首先使用 Float.int() 函数功能将其转换为整数，然后再通过 elementAt 函数进行搜索；
- (5) 如果 separator 是空字符串，则返回无效值 invalid。

下面给出了使用 elementAt 函数的几个例子：

```
var a = "My name is Joe; Age 50;";
var b = String.elementAt(a, 0, " ");      // 返回序号为 0 即第 1 个由空格分隔的元素，故 b = "My"
var c = String.elementAt(a, 14, ";");     // 由分号分隔，序号为 14 的元素不存在，所以 c = ""
```

```
var d = String.elementAt(a, 1, ";");    // 返回由分号分隔，序号为 1 的元素，所以 d = " Age 50"
```

9.3.9 removeAt 函数

removeAt 函数的功能是在给定的字符串中，删除在序号 index 处由分隔符 separator 所确定的元素。它的语法格式为：

```
String.removeAt(string, index, separator)
```

它有 3 个参数，string 参数是给定的字符串，index 参数用于指定被删除元素的序号，separator 参数定义的是用来确定元素分割的分割符。

如果用户给出的 index 比 0 小，那么 removeAt 函数将把 index 作为 0 对待。如果 index 多于元素的个数，那么 removeAt 函数就会把最后一个元素删除掉。如果 string 或 separator 是空字符串，那么 removeAt 函数就返回一个空字符串。

如果 index 是浮点数，则 WMLScript 首先使用 Float.int() 函数将 index 转换成整数，然后再通过 removeAt 函数删除指定序号处的元素。

如果给定的 3 个参数不符合要求，则 removeAt 函数就会返回无效值 invalid。

下面我们给出字符串变量 a，利用 removeAt 函数对它进行不同的删除操作后，可以得到相应的返回结果：

```
var a = "A  A;  B  C  D";  
var s = "  ";  
var b = String.removeAt(a, 1, s);    // 删除后结果为 b = "A  B  C  D"  
var c = String.removeAt(a, 0, ";");  // 删除后结果为 c = "  B  C  D"  
var d = String.removeAt(a, 14, ";"); // 删除后结果为 d = "A  A"
```

9.3.10 replaceAt 函数

replaceAt 函数用于将给定字符串中，位于序号 index 处的元素替换成新的 element 元素，该序号是通过分隔符 separator 分隔后形成的各元素序号。replaceAt 函数的语法格式为：

```
String.replaceAt(string, element, index, separator)
```

它有 4 个参数, string 参数是给定的字符串, element 参数是用于替换的字符串元素, index 参数是拟被替换元素的序号, separator 参数是分隔给定字符串的分隔符。

replaceAt 函数工作时, 如果给定的 index 比 0 小, 那么它就替换第 1 个元素。如果给定的 index 比元素的个数还要大, 那么就替换最后 1 个元素。如果 string 是空字符串, 那么就返回 1 个包含 element 的字符串。而如果 separator 是空字符串, 那么就会返回无效值 invalid。

特别地, 如果 index 是浮点数, 那么 WMLScript 将首先使用 Float.int() 函数将其转换为整数, 然后再使用 replaceAt 函数进行替换。

```
var a = "B C; E";  
var s = " ";  
var b = String.replaceAt(a, "A", 0, s);      // b = "A C; E"  
var c = String.replaceAt(a, "F", 5, ";");    // c = "B C;F"
```

9.3.11 insertAt 函数

insertAt 函数用于在给定字符串的序号为 index 的字符位置上, 再插入一个由分隔符 separator 指定分割的元素 element, 然后将插入后形成的新字符串返回。如果 index 比 0 小, 则 insertAt 函数就取 index 为 0。如果 index 比给定字符串中已经有的元素个数还多, 那么就将 element 插入在原字符串的尾部。如果给定的字符串 string 是空字符串, 那么 insertAt 函数就会返回 1 个包含 element 的字符串。

insertAt 函数的语法格式如下:

```
String.insertAt(string, element, index, separator)
```

它有 4 个参数, string 参数是给定的原始字符串, element 参数指定了将要插入的元素, index 参数是要插入元素的序号位置, separator 参数是分割原始串使用的分割符。

如果 index 参数的值是浮点数, 则 WMLScript 会使用 Float.int() 函数将其首先转换成整数, 然后再进行插入操作。

如果 separator 参数值为空字符串, 那么 insertAt 函数就会返回空字符串。如果用户提供的 4 个参数不符合要求, 则 insertAt 函数就会返回无效值 invalid。

作为举例, 我们给出使用 insertAt 函数的几条语句:



```
var a = "B C; E";  
var s = " ";  
var b = String.insertAt(a, "A", 0, s);      // 插入 A , 得 b = "A B C; E"  
var c = String.insertAt(a, "X", 3, s);      // 插入 X , 得 c = "B C; E X"  
var d = String.insertAt(a, "D", 1, ";");    // 插入 D , 得 d = "B C;D; E"  
var e = String.insertAt(a, "F", 5, ";");    // 插入 F , 得 e = "B C; E;F"
```

9.3.12 squeeze 函数

`squeeze` 函数用于去掉给定字符串中的连续空格,并将去掉空格的字符串返回。它的语法格式为:

```
String.squeeze(string)
```

它只有 1 个参数 `string`,取值需为字符串或能够合法转换为字符串的数据,返回结果为字符串。如果给定的参数值不满足要求,则返回无效值 `invalid`。

下面的例子中,字符串 `b` 中含有许多空格,使用 `squeeze` 函数我们就可以去掉这些空格:

```
var a = "Hello";  
var b = " Bye Jon . See you! ";  
var c = String.squeeze(a);                // 结果为 c = "Hello";  
var d = String.squeeze(b);                // 结果为 d = " Bye Jon . See you! "
```

9.3.13 trim 函数

`trim` 函数用于裁剪给定字符串中前面和后面的所有空格,其语法格式为:

```
String.trim(string)
```

它的参数 `string` 也需取值为字符串或能够合法地转换为字符串的数据,返回结果为字符串。如果给定的参数不满足要求,则返回无效值 `invalid`。

下面的例子中,字符串 `b` 中前、后都含有许多空格,使用 `trim` 函数我们就可以去掉这些空格:

```
var a = "Hello";  
var b = "  Bye  Jon . See you!  ";  
var c = String.trim(a);           // 结果为 c = "Hello";  
var d = String.trim(b);           // 结果为 d = "Bye  Jon . See you!"
```

可以看到，trim 函数与 squeeze 函数是有很大的区别的。前者只能去掉字符串中前、后的空格，而不能去掉字符串中字符间的空格。后者可以去掉字符串中所有的空格，而不论这些空格位于什么地方。

9.3.14 compare 函数

compare 函数用于返回给定两个字符串 string1 和 string2 比较后的大小关系。比较操作是按照 WMLScript 内部定义的字符集编码关系进行的。如果 string1 小于 string2 则返回 -1；如果 string1 等于 string2 则返回 0；如果 string1 大于 string2 则返回 1。compare 函数的语法格式为：

```
String.compare(string1, string2)
```

如果给定的两个参数 string1 和 string2 不符合字符串的比较原则，即 compare 函数对它们无法进行比较，则返回无效值 invalid。

下述几条语句我们给出了使用 compare 函数比较字符串的例子：

```
var a = "Hello";  
var b = "Hello";  
var c = String.compare(a, b);           // 二者相等，所以 c = 0  
var d = String.compare("Bye", "Jon");   // B 排在 J 的前面，B 的编码小，所以 d = -1  
var e = String.compare("Jon", "Bye");   // e = 1(与上一语句相反)
```



9.3.15 toString 函数

toString 函数的功能是将给定的数值转换为与它对应的字符串。转换时将遵循 WMLScript 有关不同数据类型数据转换的规则。toString 函数的语法格式为：

```
String.toString(value)
```

对于任何有效数据类型的参数 value 值，toString 函数都可将其转换为字符串。如果 value 取值为 invalid，则会返回无效值 invalid。

下面的例子说明了 toString 函数的功能及返回结果：

```
var a = String.toString(26);           // a = "26"
var b = String.toString(true);          // b = "true"
var c = String.toString("58true");      // c = "58true"
```

9.3.16 format 函数

format 函数的功能是将给定的 value 值利用给定的 format 方式转换为字符串。其语法格式为：

```
String.format(format, value)
```

其中 format 参数只能包含 1 个指定的格式化方式。如果用户利用它指定了多于 1 个的格式化方式，format 函数则只使用第 1 个格式化方式，其余的格式将用空字符串代替。

format 参数可以包含的格式化字符串的一般形式为：

```
% [width] [.precision] type
```

其中的 width、precision 和 type 参数的取值及功能说明如下：

(1) width。它是一个非负整数，用于控制输出的最少字符数，通常是整数部分。如果输出的 value 比指定的宽度还小，那么就在左边补空格输出。如果不指定宽度 width，那么 value 所有的字符都会输出出来。如果 width 少于需要输出的数据的长度，它也不会截断数据也不会把数据四舍五入后输出，而是把数据原样输出。

(2) precision。用于指定一个非负的整数，前面跟有句点(.)号。它的作用是控制输出数

据的精度。共有以下几种情况：

d. 用于指定最少被输出的数字的数目。如果可供输出的数字不足，即数字个数小于 d，那么输出时就从左边补 0。当 value 超过精度的时候，value 就只会输出满足精度要求的部分。默认的精度要求是 1，即 d=1。如果精度指定为 0 并且 value 能够被转换为 0，那么输出时将输出 1 个空字符串("")。

f. 用于指定小数部分输出的数字个数。如果有小数点，则至少有 1 个数出现在小数点之前。超过 f 的要求时，数值将被四舍五入到合适的数字数目后再行输出。默认情况下的精度是 6，即 f=6。如果精度是 0 或者有 1 个小数点，但没有数值，那么此时输出时小数点就不会输出出来。当小数点后的数字的数目少于精度要求时，format 函数将使用字符 0 来填补空格部分。例如，String.format("%2.3f",1.2)的结果将是" 1.200"。

s. 用于指定被格式化的最大字符数目。默认情况下所有的字符都会被输出来。当 width 比精度还大的时候，format 函数将忽略 width 的限制而直接输出字符。

(3) type。它是不可缺少的格式化参数，需出现在 width 或 precision 选项的后面。type 决定了 value 将被 format 函数进行哪一种解释，它可以取 d、f、s 三种值，代表的含义如下：

d. 代表按照整数来处理 value 值，其输出结果是[-]dddd，其中 dddd 表示一个或更多的数字。

f. 代表按照浮点数来处理 value 值，其输出结果是[-]dddd.dddd，前面的符号取决于数值的正、负。小数点后的位数取决于要求的精度。

s. 代表按照字符串来处理 value 值，并根据 width 的输出精度(即字符个数)要求输出字符。特别地，字符串中的百分号(%)可以使用两个百分号即%%来表示。

只要给出的 format 和 value 有意义，format 函数就可以输出有效的结果，结果是字符串形式的。如果 format 指定的格式化格式非法或 value 值无法处理，那么 format 函数将输出无效值 invalid。

作为举例，我们给出使用 format 函数的多个例子，从中可以进一步了解该函数的参数用法及处理规律：

```
var a = 45;
var b = -45;
var c = "now";
var d = 1.2345678
```

```

var e = String.format("e: %6d", a);           // e = "e: 45"
var f = String.format("%6d", b);              // f = "   -45"
var g = String.format("%6.4d", a);            // g = "   0045"
var h = String.format("%6.4d", b);            // h = "   -0045"
var i = String.format("Do it %s", c);          // i = "Do it now"
var j = String.format("%3f", d);              // j = "1.234567"
var k = String.format("%10.2f%%", d);          // k = " 1.23%"
var l = String.format("%3f %2f.", d);          // l = "1.234567 ."
var m = String.format("%.0d", 0);             // m = ""
var n = String.format("%7d", "Int");          // n = invalid
var o = String.format("%s", true);            // o = "true"

```

9.4 URL 库及其函数

WMLScript 的 URL 库包含一系列与绝对 URL 地址和相对 URL 地址操作相关的函数，利用它们可以实现对外部文件、外部函数等外部资源的合法调用。URL 库函数主要包括 escapeString、getBase、getFragment、getHost、getScheme、getPath、getPort、getQuery、getReferer、getParameters、isValid、loadString、resolve 和 unescapeString，本节我们就讲解这些库函数的功能和用法。

9.4.1 isValid 函数

isValid 函数用于判断给定 URL 地址的格式是否正确。如果 URL 地址合法，则该函数会返回 true，否则返回 false。isValid 函数可以判断绝对和相对 URL 地址的格式，判断时相对 URL 不必转化为绝对 URL，而是可以直接判断。isValid 函数的语法格式为：

```
URL.isValid(url)
```

它的参数 url 是字符串形式的 URL 地址，返回值为布尔型的 true/false。如果给出的参数内容不符合要求，则 isValid 函数会返回无效值 invalid。

WMLScript 中合法的 URL 地址格式为：

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

凡不符合该格式的 URL 地址，isValid 函数均会判断其为 false 或 invalid。

下面我们给出使用 isValid 函数判断 URL 地址的几个例子：

```
var a = URL.isValid("http://w.acmhst.com/script#func( )");      // a = true
var b = URL.isValid("../common#test( )");                      // b = true
var c = URL.isValid("experimental?://www.host.com/cont");      // c = false
```

9.4.2 getScheme 函数

getScheme 函数用于寻找并返回给定 URL 地址中的 Scheme 部分。如果给定 URL 地址中没有 Scheme 部分,则该函数返回空字符串。如果给定的 URL 地址无效,则会返回 invalid。getScheme 函数也支持绝对和相对 URL 地址,操作时也不会把相对 URL 地址转化为绝对 URL 地址,而是可以直接操作。该函数的语法格式如下：

URL.getScheme(url)

它的参数 url 必须是字符串形式的 URL 地址。下面是使用 getScheme 函数的几个例子：

```
var a = URL.getScheme("http://w.h.com/path#frag");              // a = "http"
var b = URL.getScheme("w.h.com/path#frag");                    // b = ""
var c = URL.getScheme("http://www.mywap.com/script#func( )");  // c = "http"
```

9.4.3 getHost 函数

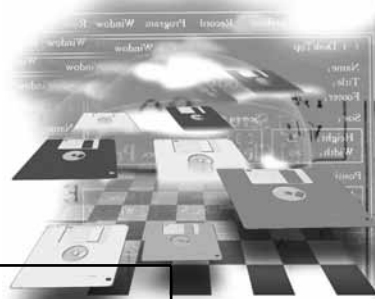
getHost 函数用于获取并返回给定 URL 地址中的主机名,即 Host 部分。如果 URL 地址中没有给出主机名,则该函数返回空字符串。getHost 函数也支持绝对和相对 URL 地址,操作时也不进行相对 URL 地址向绝对 URL 地址的转换。getHost 函数的语法格式为：

URL.getHost(url)

它的参数 url 必须为字符串形式的 URL 地址,返回值为字符串型的主机名。如果 getHost 函数无法解析给出的 URL 地址,那么它将返回无效值 invalid。

使用 getHost 函数的例子如下：

```
var a = URL.getHost("http://w.h.com/path#frag");               // a = "w.h.com"
var b = URL.getHost("path#frag");                              // b = ""
```



```
var c = URL.getHost("http://www.mywap.com/script#func( )"); // c = "www.mywap.com"
```

9.4.4 getPort 函数

getPort 函数用于获取并返回给定 URL 地址中的通信端口号，即 port 部分。如果 URL 地址中没有给出端口号，则该函数返回空字符串。getPort 函数也支持绝对和相对的 URL 地址，操作时无需转换相对 URL 地址为绝对 URL 地址。它的语法格式为：

```
URL.getPort(url)
```

它的参数 url 取值应为字符串形式的 URL 地址，函数返回值为字符串形式的端口号。如果 URL 地址中的端口号无法被 getPort 函数所解析，那么该函数就会返回无效值 invalid。getPort 函数的使用十分简单，下述几条语句就是用它来获取端口号的例子：

```
var a = URL.getPort("http://w.h.com:80/path#frag");           // a = "80"
var b = URL.getPort("http://www.mywap.com/path#frag");        // b = ""
var c = URL.getPort("http://www.mywap.com:8080/script#func( )"); // c = "8080"
```

9.4.5 getPath 函数

用于获取和返回给定 URL 地址中的路径(path)部分。如果 URL 地址中没有给出路径，那么该函数就会返回空字符串。getPath 函数也支持绝对和相对的 URL 地址，操作时不必将相对 URL 地址转化为绝对 URL 地址。它的语法格式为：

```
URL.getPath(url)
```

它的参数 url 取值也应为字符串形式的 URL 地址，函数返回值为字符串形式的路径。如果 getPath 函数无法解析 URL 地址中的路径，那么就会返回无效值 invalid。

```
var a = URL.getPath("http://www.mywap.com/home/sub/comp#frag");
// 返回的路径为 a = "/home/sub/comp"

var b = URL.getPath("../home/sub/comp#frag");           // 返回的路径为 b = "../home/sub/comp"
var c = URL.getPath("http://www.mywap.com:8080/program/WML/Script/script#func( )");
// 返回的路径为 c = "/program/WML/Script/script"
```

9.4.6 getParameters 函数

getParameters 函数用于获取和返回给定 URL 地址中的参数部分,即 params 部分。如果 URL 地址中没有指定相关参数(params),那么函数就会返回空字符串。该函数也同时支持绝对和相对的 URL 地址,操作相对 URL 地址无需将它转化为绝对 URL 地址。getParameters 函数的语法格式如下:

```
URL.getParameters(url)
```

它的参数 url 取值也应为字符串形式的 URL 地址,函数返回值为字符串形式的参数(params)。如果给定的 URL 地址无效,则 getParameters 函数返回无效值 invalid。

作为举例,我们给出使用 getParameters 函数获取参数的例子:

```
var a = URL.getParameters("http://www.waphome.com/script;3;2?x=1&y=3");    // a = "3;2"
var b = URL.getParameters("../script;3;2?x=1&y=3");                        // b = "3;2"
var c = URL.getParameters("http://www.mywap.com:8080/script#func()");      // c = ""
```

9.4.7 getQuery 函数

getQuery 函数用于获取和返回给定 URL 地址中的 query 部分。如果 URL 地址中没有 query 部分,则函数返回空字符串。该函数也同时支持绝对和相对的 URL 地址,操作时相对 URL 地址不必转化为绝对 URL 地址。getQuery 函数的语法格式为:

```
URL.getQuery(url)
```

它的参数 url 取值应为字符串形式的 URL 地址,函数返回值为字符串形式的 query 部分。如果给定的 URL 地址无效,则 getQuery 函数返回无效值 invalid。

例如,下面就是使用 getQuery 函数的例子:

```
var a = URL.getQuery("http://www.waphome.com/script;3;2?x=1&y=3");    // a = "x=1&y=3"
var b = URL.getQuery("../script;3;2?x=1&y=3");                        // b = "x=1&y=3"
var c = URL.getQuery("http://www.mywap.com:8080/script#func()");      // c = ""
```




9.4.8 getFragment 函数

用于获取和返回给定 URL 地址中的 fragment 部分。如果其中没有 fragment 部分,则函数返回空字符串。getFragment 函数也支持相对和绝对的 URL 地址,并且操作时相对的 URL 也不进行绝对 URL 的转换。getFragment 函数的语法格式如下:

```
URL.getFragment(url)
```

它的参数 url 应取字符串形式的 URL 地址,函数返回值为字符串形式的 fragment 部分。如果给定的 URL 地址无效,则 getFragment 函数返回无效值 invalid。

使用 getFragment 函数获取 fragment 部分的例子如下:

```
var a = URL.getFragment("http://www.waphome.com/cont#frag");           // a = "frag"
var b = URL.getFragment("../script;3;2?x=1&y=3");                       // b = ""
var c = URL.getFragment("http://www.mywap.com:8080/script#func( )");    // c = "func( )"
```

9.4.9 getBase 函数

getBase 函数用于获取和返回当前 WMLScript 文件的不含 fragment 部分的绝对 URL 地址。它没有参数,返回值为字符串形式的绝对 URL 地址。其语法格式为:

```
URL.getBase( )
```

例如,当前测试的 WMLScript 文件位于服务器 www.host.com 的 test 目录中,则使用如下的 getBase 函数测试时,可返回结果 a = "http://www.host.com/test":

```
var a = URL.getBase( );           // 结果为 a = "http://www.host.com/test"
```

9.4.10 getReferer 函数

getReferer 函数用于获取和返回调用当前 WMLScript 文件的最小的相对 URL 地址。如果文件调用时没有指定 URL 地址,那么该函数返回空字符串。其语法格式为:

```
URL.getReferer( )
```

它没有参数,返回值为字符串形式的相对 URL 地址。

例如，调用的当前 WMLScript 文件位于服务器 www.host.com 的 test 目录中，文件名为 app.wml，则使用 getReferer 函数测试时则可返回结果 referer = "app.wml"：

```
var referer = URL.getReferer();           // 结果为 referer = "app.wml"
```

9.4.11 resolve 函数

resolve 函数根据给定的 baseUrl 和 embeddedUrl 两个 URL 地址的参数值，组合生成一个绝对 URL 地址，并返回该地址。如果参数 embeddedUrl 的值已经是一个绝对的 URL 地址，则函数直接把 embeddedUrl 的值作为结果返回。resolve 函数的语法格式如下：

```
URL.resolve(baseUrl, embeddedUrl)
```

它的两个参数 baseUrl 和 embeddedUrl 都是字符串形式的 URL 地址，返回值结果也为字符串形式的 URL 地址。如果参数不合法，则函数返回无效值 invalid。

下面就是应用 resolve 函数合成一个有效的绝对 URL 地址的例子：

```
var a = URL.resolve("http://foo.com/", "foo.vcf");           // 结果为 a = "http://foo.com/foo.vcf"
```

9.4.12 escapeString 函数

escapeString 函数的功能是把给定字符串 string 里面的特殊字符进行转义序列的重新编码处理。这种字符即 WML 和 WMLScript 中所指的转义字符，包括 ASCII 码中的控制字符和空格，以及一些保留字作用的特殊符号：“;”、“/”、“?”、“:”、“@”、“&”、“=”、“+”、“\$”、“,”、“{”、“}”、“|”、“\”、“^”、“[”、“]”、“'”、“<”、“>”、“#”、“%”、“ ”等。

escapeString 函数的语法格式为：

```
URL.escapeString(string)
```

它仅对所给 URL 地址中的转义字符进行转换，而不进行 URL 地址的解析和定位。如果 URL 地址中包含了无法进行转换的无效字符，则函数返回无效值 invalid。

例如，使用 escapeString 函数进行下述转换处理时，我们可以得到 a = "http%3a%2f%2fw.h.com%2fdck%3fx%3d%7f%23crd" 的转换结果：

```
var a = URL.escapeString("http://w.h.com/dck?x=\u007f#crd");
```

// 结果为 a = "http%3a%2f%2fw.h.com%2fdck%3fx%3d%7f%23crd"

9.4.13 unescapeString 函数

与 escapeString 函数的作用相反,unescapeString 函数可以将给定的转义字符串 string 进行还原。而且是只做字符的转换处理,不对 URL 地址进行其他任何解析处理。unescapeString 函数的语法格式为:

```
URL.unescapeString(string)
```

转义字符还原后,unescapeString 函数返回的结果是字符串形式的 URL 地址。如果转义字符串 string 中包含了不合法的转义字符,该函数返回无效值 invalid。

例如,下面我们给出了已经转义的字符串 a,利用 unescapeString 函数对它进行还原后,可以得到原来的 URL 地址:

```
var a = "http%3a%2f%2fw.h.com%2fdck%3fx%3d%7f%23crd";  
var b = URL.unescapeString(a);           // 结果为 b = "http://w.h.com/dck?x=12#crd"
```

9.4.14 loadString 函数

该函数用于获取和返回一个由给定参数 url 与 contentType 所代表的内容形式,其语法格式为:

```
URL.loadString(url, contentType)
```

其中参数 url 是给定的字符串形式的绝对 URL 地址,参数 contentType 是给定的字符串形式的拟采用的内容类型。

特别地,给定的 contenttype 必须遵循下述规则,否则会导致错误。

(1) 只能指定一个 contenttype。即整个字符串只允许有一个内容类型(contenttype),并且不能有多余的前缀和后缀。

(2) 类型必须是 text,但是子类型可以是其他的类型。因此,类型的前缀必须是"text/"。

loadString 函数的运行行为是这样的:首先装入指定的 contentType 和 url,同时将其其他缺省的属性也一并装入进来,然后根据装入情况进行相应处理。如果装入成功,则返回符



合给定 contenttype 的内容,并且将 content 转换为字符串。如果装入失败,或者返回的 content 内容是错误的,那么就返回一个指定的错误代码。如果使用 HTTP 或 WSP 协议,则返回 HTTP 的错误代码。

如果用户给定的是一个无效的 contenttype,那么 loadString 函数就会返回无效值 invalid。下面的简单例子给出了使用 loadString 函数的一般方法:

```
var myUrl = "http://www.host.com/vcards/myaddr.vcf";  
myCard = URL.loadString(myUrl, "text/x-vcard");
```

9.5 WMLBrowser 库及其函数

WMLBrowser 库提供了使用 WMLScript 操作 WML 卡片及 WML 浏览器的各种功能函数,这些函数为 WMLScript 和 WML 的结合使用提供了很好的支持。不过,如果所用系统不支持 WML 浏览器,或 WMLScript 的解释器不是由 WML 浏览器所激活的,那么 WMLBrowser 库的所有函数均不能进行有效的操作和处理,只能返回无效值 invalid。

WMLBrowser 库函数主要包括 getCurrentCard、getVar、go、newContext、prev、refresh 和 setVar 函数,下面我们就详细介绍这些函数的功能与用法。

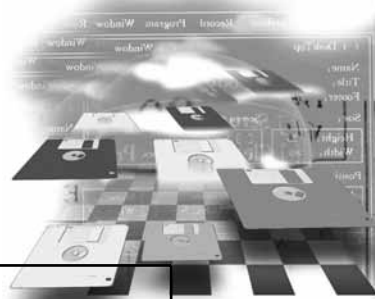
9.5.1 getVar 函数

getVar 函数用于获取和返回给定变量名的变量值,该变量名由参数 name 给定,变量名必须遵循 WML 有关变量命名的语法和规定。如果参数指定的变量不存在,那么函数就会返回空字符串。getVar 函数的语法格式如下:

```
WMLBrowser.getVar(name)
```

其中的参数 name 必须是字符串形式表示的变量名,变量值也将以字符串的形式返回。如果变量名的语法不正确,那么函数就会返回无效值 invalid。

例如,在下面的几行程序中,变量 name 被赋值"Good!",所以当使用 getVar 函数测试它的变量值的时候,会得到"Good!"的结果。



```
var name = "Good!";  
var a = WMLBrowser.getVar("name");           // 返回变量 name 的值，故 a = "Good!"
```

9.5.2 setVar 函数

setVar 函数的功能是，在当前浏览器运行的程序中，就给定两个参数 name 和 value 进行检验和操作，如果 name 参数代表的变量名确实存在，那么就将 value 参数代表的变量值赋给 name 参数代表的变量，然后返回 true；否则，就不赋值，同时返回 false。setVar 函数的语法格式如下：

```
WMLBrowser.setVar(name, value)
```

其中参数 name 和 value 所代表的变量名及变量值都必须符合 WML 的语法规则，变量值必须是 XML 合法的 CDATA 型数据。如果给定的变量名不符合语法规则或者数值类型不正确，那么函数就会返回无效值 invalid。

例如，当前浏览器运行的 WML 程序中，我们已经定义了 name 变量，即程序中确实存在该变量，所以当使用 setVar 函数进行检验和操作时，就会得到变量名为真(true)的判断，同时会把给定的参数值 "Mary" 赋给该变量：

```
var a = WMLBrowser.setVar("name", "Mary");           // 返回结果为 a = true
```

setVar 函数赋值的条件是检验的变量名确实存在，而不管该变量是否已经赋值，也不管该变量参与何种运算；而且一旦赋值，该变量在程序以后的运行中均采用新赋的值进行操作或运算。

9.5.3 go 函数

go 函数用于装入由给定参数 url 所指定 URL 地址的内容。它的功能与 WML 语言的 go 任务的功能完全相同。该 URL 的装入是在 WML 浏览器接收到 WMLScript 解释器调用操作的结束信息后开始的，即先由 WMLScript 解释器调用执行当前的 go 函数，执行完后向 WML 浏览器发出信息，浏览器随后即装入 go 函数指定 URL 地址的内容。如果装入成功，则返回一个空字符串。如果给定的参数 url 是个空字符串，那么浏览器将无法装入指定内容，同时也返回一个空字符串。go 函数的语法格式如下：

```
WMLBrowser.go(url)
```

其中的参数 url 需取字符串形式的 URL 地址。如果给定的 URL 地址不合法，则 go 函数就会返回无效值 invalid。例如，下面的例子即可使用 go 函数装入指定 URL 地址 <http://www.host.com/loc/app.dck#start> 的内容：

```
var card = "http://www.host.com/loc/app.dck#start";  
WMLBrowser.go(card);
```

9.5.4 prev 函数

prev 函数的功能是在当前程序执行的过程中，使 WML 浏览器返回到显示的上一个卡片的内容，它与 WML 语言的 prev 任务具有相同的功能。这里的卡片装入也是在 WML 浏览器接收到 WMLScript 解释器调用操作的结束信息后开始的，即先由 WMLScript 解释器调用执行当前的 prev 函数，执行完后向 WML 浏览器发出信息，浏览器随后装入 prev 函数所指定的上一卡片的内容。prev 函数的语法格式为：

```
WMLBrowser.prev()
```

它没有参数，但它有明确的操作对象，即 WML 浏览器显示的上一个卡片。与 go 函数一样，如果装入成功，prev 函数也返回一个空字符串。如果 WML 浏览器前面还没有显示过卡片，即上一卡片的 URL 地址为空，那么浏览器将不执行 prev 函数的操作，也不试着装入上一卡片的内容，同时返回一个空字符串。

其实，prev 函数和 go 函数的调用是相互覆盖的，它们可以在程序将控制权返回给浏览器之前反复地调用、装入指定 URL 地址的内容，go 函数往前调用，而 prev 函数往后调用。

9.5.5 newContext 函数

这是一个比较简单的函数，它的功能是清理当前 WML 浏览器中的内容并返回空字符串，与 WML 语言的 Newcontext 属性具有相同的功能。newContext 函数的语法格式为：

```
WMLBrowser.newContext()
```

如果当前 WML 的浏览器存在某些问题而无法清理，函数会返回无效值 invalid。

例如，下面使用 newContext 函数的语句即可清理当前的 WML 浏览器内容：

```
WMLBrowser.newContext();
```



9.5.6 getCurrentCard 函数

getCurrentCard 函数的功能是获取并返回当前浏览器正在运行的卡片的最小的相对路径，其语法格式如下：

```
WMLBrowser.getCurrentCard()
```

它没有参数，返回值为字符串形式的相对路径。不过，如果与当前路径对应的目录下没有当前卡片，函数会返回无效值 invalid。

例如，假设当前 WML 浏览器中显示的是 <http://www.mywap.com/script/deck#input> 中的卡片，则使用 newContext 函数可以获得当前卡片的最小相对路径"deck#input"：

```
var a = WMLBrowser.getCurrentCard();           // 返回结果为 a = "deck#input"
```

9.5.7 refresh 函数

refresh 函数的功能是强制 WML 浏览器更新内容并返回空字符串。在这一点上它与 WML 语言的 refresh 任务的功能是完全相同的。其语法格式为：

```
WMLBrowser.refresh()
```

它也没有参数，当函数生效时将返回一个空字符串，否则，如果浏览器内容无法更新或其中没有显示卡片，则会返回无效值 invalid。

refresh 函数的使用方法十分简单，下面的自定义函数中就利用了 refresh 函数进行浏览器内容的更新：

```
function convert2Peso() {  
    var dollars = WMLBrowser.getVar("amount");  
    var dol2peso = 10.2;  
    var newAmt = dollars*dol2peso;  
    WMLBrowser.setVar("amount", newAmt);  
    WMLBrowser.refresh();  
}
```

9.6 Dialogs 库及其函数

Dialogs 库提供了 3 个用于用户交互界面操作与处理的功能函数，即 prompt、confirm 和 alert，下面我们就讲解这 3 个函数的功能和用法。

9.6.1 prompt 函数

prompt 函数的功能是显示给定的信息并提示用户输入，当用户输入有效的内容后再将用户输入作为结果返回给调用方。该函数的语法格式为：

```
Dialogs.prompt(message, defaultInput)
```

它有两个参数，其中 message 参数是个字符串，它是用户给定的用于显示的信息；defaultInput 参数用于接受用户输入，同时它包含了一个初始的数值，如果用户不输入，函数将把该初始值返回，否则，就返回用户的输入值。

如果参数内容不符合 WMLScript 的要求，则 prompt 函数返回无效值 invalid。

看下面使用 prompt 函数的简单例子：

```
var a = "13098762345";  
var b = Dialogs.prompt("Phone number: ",a);  
// 如果用户输入了 13012348765，则返回结果为 b=13012348765
```

它在浏览器上显示的结果是 :Phone number: 13098762345 ,如果用户想输入新电话号码，则可在现有号码"13098762345"处直接输入，新输入的号码将替换原有的号码，prompt 函数在最后还会将新输入的号码返回。

9.6.2 confirm 函数

confirm 函数用于提供确认功能，它首先显示给定的 message 参数所包含的信息，并提供两个可选择的回答“ok”和“cancel”，然后等待用户选择其中的一个。如果用户选择“ok”则返回 true；否则，如若选择“cancel”则返回 false。该函数的语法格式如下：

```
Dialogs.confirm(message, ok, cancel)
```




它有 3 个参数，其中参数 `message` 是欲显示的字符串形式的提示信息；参数 `ok` 也是用于显示的一个字符串信息，它可以是“OK”，也可以是其他表示 OK 意义的文本信息，如“I Agree”、“I Like”等等；同样，参数 `cancel` 也是用于显示的字符串信息，可以是“Cancel”文本，也可以是其他表示 Cancel 意义的文本信息。

正常情况下，`confirm` 函数的返回结果是布尔型的 `true` 或 `false`。如果用户选择错误或拟显示的信息非法，则函数返回无效值 `invalid`。

下面就是使用 `confirm` 函数的简单例子：

```
function onAbort() {  
    return Dialogs.confirm("Are you sure?", "Yes", "No");  
};
```

它显示的结果是：Are you sure? Yes No

其中的“ Yes ”、“ No ”是可以选择的，且 `confirm` 函数将根据用户选择的结果返回 `true` 或 `false`。

9.6.3 alert 函数

`alert` 函数的功能是显示给定的 `message` 信息给用户，并等待用户的确认，最后返回空字符串作为结束。其语法格式为：

```
Dialogs.alert(message)
```

如果用户操作错误或浏览器显示有问题，则 `alert` 函数会返回无效值 `invalid`。

下面就是使用 `alert` 函数的简单例子：

```
function testValue(textElement) {  
    if (String.length(textElement) > 8) {  
        Dialogs.alert("Enter name < 8 chars!");  
    };  
};
```

它的作用是，对用户输入名字的长度进行判断，该名字存放在变量 `textElement` 中。如果多于 8 个字符，则使用 `alert` 函数显示输入名字必须少于 8 个字符的信息：“Enter name < 8 chars!”。

9.7 WMLScript 非标准库及其库函数

以上各节我们介绍的都是 WMLScript 的标准库函数，事实上，WMLScript 还有一些非标准库函数。这些函数通常是由不同的 WAP 设备厂家制定并提供的，而且因厂家不同而不同，所以称为“非标准”的库函数。例如，Nokia 提供了 Debug 库函数，Phone.com 提供了 Console 库函数，等等。虽然这些库的名称不同，但涉及的功能函数却是基本一致的，它们主要是为用户操作提供一些辅助功能。下面我们就以 Debug 库为例介绍 3 个比较典型的非标准库函数。

9.7.1 openFile 函数

openFile 函数的功能是为文件读、写或追加等操作而打开一个文件。它的语法格式为：

```
Debug.openFile(fileName, mode)
```

它有两个参数，其中 fileName 参数通常以字符串的形式来指定要打开的文件名，可以包含文件所在的路径名称；mode 参数用于指定打开文件后的操作方式，共 3 种选择："r"指定打开的文件用于读操作，"w"指定文件用于写操作，"a"指定文件用于追加操作。

正常情况下，openFile 函数的返回结果为空字符串。如果指定的文件不存在，或文件的操作方式不对，或文件无法操作，那么 openFile 函数都会返回无效值 invalid。

如果给定的 fileName 和 mode 参数值的形式不是字符串形式，那么 openFile 函数也会返回无效值 invalid。

下面就是使用 openFile 函数打开指定文件并进行不同操作的例子：

```
Debug.openFile("c:\\tmp\\script.debug", "r");  
Debug.openFile("c:\\tmp\\debug", "a");
```



9.7.2 closeFile 函数

openFile 函数用于打开文件，当对文件操作完毕后必须关闭文件，否则就会引起文件处理错误甚至破坏文件，那么如何关闭文件呢？方法就是使用 closeFile 函数。该函数没有参数，它用于关闭当前打开的文件，语法格式为：

```
Debug.closeFile ( )
```

它的返回值为空字符串。下述语句就是使用 closeFile 函数关闭当前文件的例子：

```
Debug.openFile("c:\\tmp\\script.debug", "r");  
.....(其他语句)  
Debug.closeFile ( );  
.....  
Debug.openFile("c:\\tmp\\debug", "a");  
.....(其他语句)  
Debug.closeFile ( );
```

9.7.3 println 函数

println 函数用于输出给定的字符串 string 中的信息。如果当前系统中存在由 openFile 函数打开的文件，则 println 函数将该信息直接写入这一打开的文件；否则，println 函数就将该信息送往标准的输出设备，如显示器、打印机等，然后再由这些设备输出给用户。该函数的语法格式如下：

```
Debug.println(string)
```

如果给定的 string 信息能够正常输出，则 println 函数返回一个空字符串；否则，如若输出信息类型不符合要求，或当前系统中不存在合法的打开文件或输出设备，则 println 函数返回无效值 invalid。

下述语句就可以利用 println 函数向当前打开的文件或输出设备中输出指定的信息：

```
Debug.println("Function f1 ENTRY ...");           // 输出"Function f1 ENTRY ..."  
Debug.println("Function f1 EXIT ...");           // 输出"Function f1 EXIT ..."
```

9.8 WML/WMLScript 应用举例

通过前面的学习,我们已经全面了解了 WMLScript 的编程语法、规则和各种库函数,为进一步加强大家对 WMLScript 编程的认识,我们本节先给出一个 WMLScript 库函数的使用举例,然后给出 Phone.com 提供的几个综合利用 WML、WMLScript 开发的应用实例。

9.8.1 WMLScript 库函数应用举例

我们利用有关库函数设计了一个计算抵押贷款每月偿还金的外部函数。该函数名为 payment,它有 4 个参数:varname、principal、interest 和 num_payments,其中 varname 用于存放最后的计算结果,principal 为本金,interest 为利率,num_payments 为偿款次数。用到的库函数是:用于幂运算的 Float.pow(),用于字符串格式化输出的 String.format(),用于浏览器变量显示的 WMLBrowser.setVar()以及用于浏览器刷新的 WMLBrowser.refresh()。

程序如下:

```
extern function payment(varname, principal, interest, num_payments) {
    /*
     * 利息计算公式为:
     *
     * If (i != 0), then:
     * pmt = principal * [i * (1+i)^n / ((1+i)^n - 1)]
     *
     * If (i == 0), then:
     * pmt = principal / n
     */
    var mi = interest/1200;           // 由年度利率计算每月利率
    var payment = 0;
    if (mi != 0) {
        var tmp = Float.pow((1 + mi), num_payments);
        payment = principal * (mi * tmp / (tmp - 1));
    } else {
        if (num_payments != 0)
            payment = principal / num_payments;
    }
}
```



```

var s;
if (payment != 0)
    s = String.format("$%6.2f", payment);
else
    s = "Missing data";

/*
 * 将结果发送到浏览器显示
 */
WMLBrowser.setVar(varname, s);

/*
 * 刷新当前浏览器
 */
WMLBrowser.refresh();
};

```

9.8.2 数值范围有效性检验实例

本例主要用于说明使用 WML 和 WMLScript 语言实现数值有效性检验的具体方法。

本例共有两个程序文件：validate.wml 和 validator.wmls。其中 validate.wml 页面文件建立了 3 个卡片：一个用于让用户输入指定范围的数值，另外两个用于显示判断结果。validator.wmls 文件的程序用于判断用户输入的数值，看其范围是否有效。如果没有超出指定的范围，则向 WAP 浏览器返回数值结果，否则就向用户显示数值超出范围的错误信息。

Validate.wml 文件的程序清单如下：

```

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//PHONE.COM/DTD WML 1.1//EN" "http://www.phone.com/dtd/
wml11.dtd">

<wml>
// 初始化卡片
<card id="first">
    <onevent type="onenterforward">
        <refresh>
            <setvar name="num" value=""/>

```



```
</refresh>
</onevent>

<p>
  <do type="accept" label="valid">
    // 调用 validator.wmls 中的外部函数 validate()进行有效性检验
    <go href="validator.wmls#validate()"/>
  </do>
  Enter a whole number between 25 and 50
  <input type="text" name="num" format="*N"/>      // 输入数值
</p>
</card>

// 输入数值无效时显示的卡片
<card id="invalid">
<p>
  <do type="accept" label="again">
    <go href="#first">      // 返回 first 卡片重新输入
    <setvar name="num" value=""/>      // 返回前对变量进行重新初始化
  </go>
</do>
// 显示无效性信息
$(num) is not in the range 25 to 50
</p>
</card>

// 输入数值有效时显示的卡片
<card id="valid">
<p>
  <do type="accept" label="again">
    <go href="#first">      // 返回 first 卡片重新输入
    <setvar name="num" value=""/>      // 返回前对变量进行重新初始化
  </go>
</do>
// 显示有效性信息
```



```

$(num) is in the range 25 to 50
</p>

</card>
</wml>

```

Validator.wmls 文件的程序清单如下：

```

extern function validate(){
    var validNum = WMLBrowser.getVar("num");           // 获取变量值
    var validNumAsInt = Lang.parseInt(validNum);        // 数据类型转换
    var max = 50;
    var min = 25;
    if (validNumAsInt){
        if ((validNumAsInt < min) || (validNumAsInt > max)) {
            // 若数值范围无效则至 validate.wml 文件中的 invalid 卡片
            WMLBrowser.go("validate.wml#invalid");
        }
        else {
            // 若数值范围有效则至 validate.wml 文件中的 valid 卡片
            WMLBrowser.go("validate.wml#valid");
        }
    }
    else {
        // 数值类型及范围均无效，显示错误信息并进行变量初始化，
        // 然后返回 validate.wml 的 first 卡片重新输入
        Dialogs.alert("The number " + validNum + "is not a whole number");
        WMLBrowser.setVar("num", "");
        WMLBrowser.go("validate.wml#first");
    }
}

```

本例的主要执行步骤解释如下：

首先 validate.wml 进行卡片初始化，并完成两项任务。其一，提示用户输入一个 25 到 50 之间的整数，并接收用户输入，把数值赋给变量 num，然后调用 validator.wmls 文件中外部函数 validate()：`<go href="validator.wmls#validate()"/>`，以测试 num 的范围。其二，如果用户从另外两个卡片返回到初始化卡片，则再次对 num 变量进行初始化，以便为用户的下

一轮输入做好准备：

```
<do type="accept" label="again">
  <go href="#first">
    <setvar name="num" value=""/>
  </go>
</do>
```

如果用户输入的数值是个整数并满足范围要求，则 validate() 函数就会调用 valid 卡片显示到浏览器上，而若是整数但不满足范围要求则调用 invalid 卡片，若不是整数则调用 first 卡片重新输入：

```
if (validNumAsInt) {
  if ((validNumAsInt < min) || (validNumAsInt > max)) {
    WMLBrowser.go("validate.wml#invalid");
  }
  else { WMLBrowser.go("validate.wml#valid"); }
}
else {
  Dialogs.alert("The number " + validNum + "is not a whole number");
  WMLBrowser.setVar("num", "");
  WMLBrowser.go("validate.wml#first");
}
```

9.8.3 货币换算实例

本例使用 WML 和 WMLScript 建立了一个简单的货币换算器，同时也给出了利用 WML 和 WMLScript 实现与用户交互的基本方法。

本例有两个程序文件：currency.wml 和 converter.wmls。前者建立了 4 个卡片，用于完成货币值和类型的输入、输出；后者负责将输入的货币值转换为指定货币的值，并显示到 WAP 浏览器上。

Currency.wml 文件的程序清单如下：

```
<?xml version="1.0"?>
```




```
<!DOCTYPE wml PUBLIC "-//PHONE.COM//DTD WML 1.1//EN" "http://www.phone.com/dtd/wml11.dtd">
```

```
<wml>
  // 初始化卡片
  <card id="first" >
    <onevent type="onenterforward">
      <refresh>
        // 刷新变量值，即初始化
        <setvar name="startCur" value=""/>
        <setvar name="endCur" value=""/>
        <setvar name="cur" value=""/>
      </refresh>
    </onevent>

    <p>
      <do type="accept" label="conv">
        <go href="#destVal"/>      // 跳转到 destVal 卡片
      </do>
      Currency converter!
      Convert from?
      // 显示选单，并让用户选择换算的原货币
      <select name="startCur">
        <option value="USD">USD</option>
        <option value="DMark">D-Mark</option>
        <option value="Franc">Franc</option>
        <option value="Lira">Lira</option>
      </select>

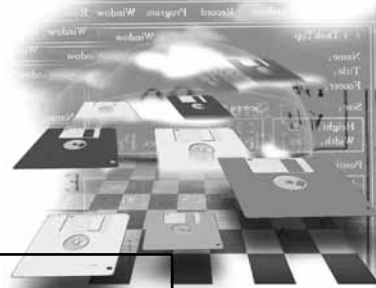
    </p>
  </card>

  // 第 2 个卡片 destVal
  <card id="destVal">
    <p>
      <do type="accept" label="dest">
        <go href="#valCard"/>      跳转到 valCard 卡片
      </do>
```

```
Convert from $(startCur) to:
// 显示选单，并让用户选择换算的目的货币
<select name="endCur">
    <option value="USD">USD</option>
    <option value="DMark">D-Mark</option>
    <option value="Franc">Franc</option>
    <option value="Lira">Lira</option>
</select>
</p>
</card>

// 第 3 个卡片 valCard
<card id="valCard">
    <onevent type="onenterforward">
        <refresh>
            // 变量 cur 和 ans 刷新，即初始化
            <setvar name="cur" value=""/>
            <setvar name="ans" value=""/>
        </refresh>
    </onevent>
</p>
<do type="accept" label="conv">
    // 调用 converter.wmls 文件中的外部函数 convert()，进行货币换算
    <go href="converter.wmls#convert()"/>
</do>
$(startCur)? (whole number)
<input type="text" name="cur" format="*N"/> // 输入要换算的原货币的货币值
</p>
</card>

// 第 4 个卡片 convD
<card id="convD">
</p>
<do type="accept" label="new">
    // 跳转到 first 卡片并进行变量初始化
    <go href="#first">
```



```

        <setvar name="startCur" value=""/>
        <setvar name="endCur" value=""/>
        <setvar name="cur" value=""/>
    </go>
</do>

<do type="options" label="again">
    <go href="#valCard"/>    //跳转到 valCard 卡片
</do>

// 显示换算结果
$(cur) in $(startCur) is $(ans) $(endCur)
</p>
</card>
</wml>

```

Converter.wmls 文件的程序清单如下：

```

extern function convert() {
    // 利用库函数获取变量值并进行类型转换
    var scur = WMLBrowser.getVar("startCur");
    var ecur = WMLBrowser.getVar("endCur");
    var cur = WMLBrowser.getVar("cur");
    var curVal = Lang.parseInt(cur);
    if(curVal){
        if(scur != ecur {
            // 若换算的货币类型不同，则进行换算
            var endval = convertVal(scur, ecur, curVal);
        }
        else { endval = cur; }    // 否则，类型相同，不进行换算
    }
    endval = String.format("%1.2f", endval);    // 对输出结果进行格式化
    WMLBrowser.setVar("ans", endval);
    WMLBrowser.go("currency.wml#convD");
}
else{
    error(cur);    // 调用错误处理函数
}
}

```



```
}

function error(badVal){           // 处理错误，并返回错误信息至 valCard 卡片显示
    Dialogs.alert("Your value " + badVal + " is bad");
    WMLBrowser.setVar("cur", "");
    WMLBrowser.go("#valCard");
}

function convertVal(scur, ecur, val) {
    // 设置货币汇率
    var usd = 1;
    var lira = 1858.22;
    var dmark = 1.6535;
    var franc = 5.5425;

    // 先一律换算为美元 USD
    if (scur == "USD"){
    } else if (scur == "Lira"){
        val = val / lira;
    } else if (scur == "DMark"){
        val = val / dmark;
    } else if (scur == "Franc"){
        val = val / franc;
    }

    // 换算为目的货币
    if (ecur == "Lira"){ // 换算为 Lira 并返回结果
        return(val*lira);
    } else if (ecur == "DMark"){ // 换算为 Deutsch Mark 并返回结果
        return(val*dmark);
    } else if (ecur == "Franc"){ // 换算为 French Franc 并返回结果
        return(val*franc);
    } else { // 换算为 US Dollar 并返回结果
        return(val);
    }
}
```



本例的主要执行步骤解释如下：

currency.wml 的 first 卡片是初始化卡，主要完成两项任务。其一，提供用户选择换算的原货币，并把选择值赋给变量 startCur，随后显示下一卡片，即 destVal 卡片；其二，用户从最后一卡片(即 convD 卡片)返回到 first 卡片，则重新对 startCur、endCur 和 cur 变量进行初始化，以为下一轮的货币换算做准备。

destVal 卡片提示用户选择换算的目的货币，并把选择值赋给变量 endCur，随后显示下一卡片，即 valCard 卡片。

第 3 个卡片 valCard 将先把变量 cur 和 ans 初始化为空字符串，并提示用户输入想要换算的货币值，并把该数值赋给变量 cur，随后调用 converter.wmls 程序文件中的外部函数 convert()：<go href="converter.wmls#convert()"/>。

第 4 个卡片 convD 用于显示货币换算的结果：\$(cur) in \$(startCur) is \$(ans) \$(endCur)，而且还对变量 startCur、endCur 和 cur 进行初始化，并返回到 first 卡片，以便让用户开始新一轮的换算操作。convD 卡片同时还给用户提供了使用相同原货币、目的货币和不同货币值的机会，其代码如下：

```
<do type="options" label="again">
  <go href="#valCard"/>
</do>
```

convert()函数初始化本地变量时使用了 WMLBrowser 库的 getVar 函数及 Lang 库的 parseInt 函数：

```
var scur = WMLBrowser.getVar("startCur");
var ecur = WMLBrowser.getVar("endCur");
var cur = WMLBrowser.getVar("cur");
var curVal = Lang.parseInt(cur);
```

convert()函数完成变量初始化后，首先检查用户输入的 curVal 值是否有效，如果无效，则调用它的内部函数 error()来通知用户，并提示用户重新输入。这段代码即为 convert()函数中的下述代码段：

```
if(curVal){
```

```
...  
}  
else{  
    error(cur);  
}
```

如果 curVal 的值有效，那么 convert()函数就开始检查用户是否选择相同的原货币和目的货币类型。如是，则无需进行换算，直接返回输入的货币值即可。否则，就责成内部函数 convertVal()根据预先给定的汇率进行货币换算。

当 convertVal()返回换算结果后，convert()函数就将换算结果变成格式化字符串显示到 WAP 浏览器上，并随后调用 currency.wml 文件中的最后一个卡片，即 convD 卡片。实现这段功能的代码如下：

```
if(curVal){  
    if(scur != ecur{  
        var endval = convertVal(scur, ecur, curVal);  
    }  
    else { endval = cur; } //no conversion  
    endval = String.format("%1.2f", endval);  
    WMLBrowser.setVar("ans", endval);  
    WMLBrowser.go("currency.wml#convD");  
}
```

9.8.4 简单动画实例

本例通过使用 WML 的 ontimer 事件来延时交替显示两幅图像，来实现一个简单的动画应用。本例同时给出了从 WML 向 WMLScript 函数传递变量值，以及获取返回值的编程方法。WAP 页面中使用的图像只能是 1 位的位图图像，格式为 WAP 专用的 wbmp 格式，有关图像格式转换的方法可参见本书第 11 章的内容。

本例有两个程序文件：animate.wml 和 animated.wmls。animate.wml 执行时首先给 time 变量赋予初始值 20，然后在用户选择“运行动画(Run Animation)”选项后对存放图像文件的变量 image 进行初始化，并赋予它第 1 张图像文件名；当 ontimer 事件激活时，它就调用



animated.wmls 文件中的 main()函数，该函数将每次使 time 变量的值减少 2，并交换 image 变量的图像文件名，随后返回新的 time 值和 image 值，并在 WAP 浏览器上显示图像。这一过程将在 ontimer 事件激活时重复一次，直到 time 变量的值最后减少到 0。

Animate.wml 文件的程序代码如下：

```
<?xml version="1.0"?>
  <!DOCTYPE wml PUBLIC "-//PHONE.COM/DTD WML 1.1//EN" "http://www.phone.com/dtd/
wml11.dtd">

  <wml>
    <card>
      <option>Run Animation          // 定义一个选项
        <onevent type="onpick">
          // 变量赋初值
          <setvar name="time" value="20"/>
          <setvar name="image" value="image1.bmp"/>
        </onevent>
      </option>
      <onevent type="ontimer">
        // 事件激活(即延时结束后)，执行 animated.wmls 文件中的外部函数 main()
        <go href="animated.wmls#main()">
        </go>
      </onevent>
      // 显示图像及提示信息
      <p align="center"></p>
      <p align="center">A simple animation</p>
    </card>
  </wml>
```

Animated.wmls 文件的程序代码如下：

```
extern function main()
{
  // 获取初始变量值
  var image = WMLBrowser.getVar("image");
  var remianingTime = Lang.parseInt(WMLBrowser.getVar("time"));
```



```
// 减少 time 的时间值，并判断值为 0 时终止执行
remainingTime -= 2;
WMLBrowser.setVar("time", remainingTime);

if (remainingTime > 0); {
    if (image == "image1.bmp") {
        WMLBrowser.setVar("image", "image2.bmp");    // 替换显示的图像文件
    } //如果当前显示的是 image1.bmp，则换为 image2.bmp
    else {
        WMLBrowser.setVar("image", "image1.bmp");
    } // 否则就换为 image1.bmp
    WMLBrowser.refresh();
} // 如果延时大于 0
}
```

本章小结

本章我们讲解了 WMLScript 语言提供的库函数，以及第三方提供的 WMLScript 非标准库函数，涉及的内容主要包括各函数的功能、用法、参数作用及取值范围等。作为 WML、WMLScript 学习的总结，本章最后还讲解了几个使用 WML、WMLScript 开发的 WAP 应用实例。本章内容较多，读者不可能一下子熟练地掌握全部内容，但希望大家能知道实现各种具体无线网络功能时所要用的库函数，这样等实际开发时就可以来快速查阅这些函数的详细内容，从而提高开发效率。



第 10 章 HDML 编程

我们知道,普通网页一般采用超文本标记语言 HTML(HyperText Markuup Language)来编写,手机等无线网络中的 WAP 网页目前大都使用 WML/WMLScript 语言来编写。事实上,WAP 网页除了可以使用 WML/WMLScript 编写外,我们还可以使用一种与 HTML 极为相似的语言来编写,这种语言就是手持标记语言 HDML(Handheld Markup Language)。由于 HTML 是一种非常简单易用的语言,所以使用 HDML 来编写网页的简单易用性就可想而知。目前 HDML 语言的最新版本为 4.0,本章我们就通过比较 HTML 和 HDML 两种语言,来讲述 HDML 语言的语法规则和编程技术。为便于大家更好地学习和接受 HDML 语言,我们先用较大的篇幅来介绍 HTML 语言的基础知识和 HTML 标签。

10.1 HTML 语言基础知识

HTML 语言编写的网页通常称为 HTML 网页、HTML 页面或 HTML 文件。产生 HTML 文件的方法主要有两种,一种是手工编制,另一种是使用诸如 FrontPage、Dreamweaver 的网页编辑器产生。一般来说,手工编制能够更灵活地应用 HTML 语言,进而能够实现更灵活的网络功能。学习手工编制 HTML 文件是很有必要的,我们这里介绍的都属于手工编制 HTML 文件的基本知识。


10.1.1 HTML 页面

HTML 页面包含两种信息:其一,页面本身的内容,如文本、图形等;其二,表示页面元素、格式、结构、超链接等的 HTML 标记(也称为标签)。编写好的 HTML 页面文件可以被浏览器执行。浏览器执行 HTML 文件遇到 HTML 标签时,它就会将页面的内容按照标签定义的格式显示出来。HTML 标签的基本格式是:

<标签名> 页面内容 </标签名>

前面的标签表示所定义的格式的开始，后面的标签表示这种格式的结束。例如“你好! ”表示用粗体字显示“你好!”。

HTML 的标签使用小于号(<)和大于号(>)括起来，即采用“<标签名>”的形式。标签分单独出现的标签和成对出现的标签两种，上面介绍的就属于成对出现的标签。大多数标签是成对出现的，由首标签和尾标签组成。首标签和尾标签又分别称为起始标签和终止标签。首标签的格式为“<标签名>”，尾标签的格式为“</标签名>”。成对标签用于规定元素所涵盖的范围，比如和标签用于界定黑体字的范围，也就是说，和之间包住的部分采用黑体字显示。单独标签的格式为“<标签名>”，它的作用是在相应的位置插入元素。如
标签表示在该标签所在位置插入一个换行符。

 需要指出的是，HTML 的标签不区分大小写，即<BODY>标签与<body>标签、<Body>标签、<boDy>标签都是同一个标签。这种规定显然为代码编写提供了方便。

当前我们常用的 Netscape Navigator 和 Internet Explorer 两种浏览器对 HTML 标签的解释并不完全一样，也就是说，同一个标签在不同浏览器中显示的结果可能不一样，两者是否能解释过去也不一定，即使同一种浏览器的不同版本也可能存在此类问题，因此编写网页时需要注意这些问题，了解浏览器的版本和要求，并据此写出满足要求的 HTML 页面。一般而言，常用的普通功能都不会出现此类现象，新的、高级的功能才可能会出现这种问题。

HTML 页面文件是纯文本文件，因此用普通的文本编辑器都可以进行编辑，只不过保存时它的扩展名必须保存成.htm 或.html。我们一般采用纯文本文档编辑器，如 Windows 中的记事本或写字板，DOS 中的文本编辑器 Edit 等来编辑 HTML 页面，并在保存时将其存为扩展名为.htm 或.html 的文档。我们不提倡使用 Word、WPS 等字处理软件来编辑 HTML 页面，因为这样会导致格式字符出错。

10.1.2 HTML 页面文件的结构

HTML 页面文件的整体结构如下：

```
<HTML>
<HEAD>
```



```
<TITLE>...</TITLE>
</HEAD>
<BODY>
.....
</BODY>
</HTML>
```

该整体结构中的 HTML 标签解释如下：

(1) <HTML>标签。每个 HTML 页面中的第一个 HTML 标签都是<HTML>，它说明文件内容是用 HTML 语言编写的，HTML 页面中的所有文本和 HTML 标签都包含在<HTML>与</HTML>开始和结尾的标签之中。

(2) <HEAD>标签。<HEAD>标签表明包含在这一标签中的文本是文件中其他文本的序言，通常只有很少的部分写在文件的<HEAD>与</HEAD>部分，我们不能把页面内容的任何文本写在这一页头部分中。

(3) <TITLE>标签。包含在<TITLE>、</TITLE>标签之间的文本称页头标题，浏览器将页头标题放在页面窗口的标题栏中。

(4) <BODY>标签。HTML 页面的内容都包含在<BODY>与</BODY>标签中，这些内容将显示在 Web 浏览器的用户区内，它们属于 HTML 页面的主体部分。在<BODY>标签中我们可以规定整个页面的一些基本属性，主要是：

- “BGCOLOR”。指定 HTML 页面的背景色。
- “TEXT”。指定 HTML 页面中文字的颜色。
- “LINK”。指定 HTML 页面中待连接超链接对象的颜色。
- “ALINK”。指定 HTML 页面中连接中超链接对象的颜色。
- “VLINK”。指定 HTML 页面中已连接超链接对象的颜色。
- “BACKGROUND”。指定 HTML 页面的背景文件。

在指定对象的颜色时，可以使用该颜色的代码或直接使用该颜色对应的英文单词，例如，我们指定 HTML 页面的背景色为绿色，可以表示为：<body BGCOLOR= “green”>，也可以表示为<body BGCOLOR= “#0080000”>。为了便于记忆，建议读者直接用相应的英文单词指定颜色。Body 属性的颜色值可以是表 10.1 所示 16 种颜色中的任何一种。



表10.1 页面主体部分的可用颜色

颜色值	颜 色	颜色值	颜 色
Black	黑 色	Red	红 色
Lime	石灰色	Maroon	栗 色
Gray	灰 色	Silver	银白色
Navy	海军蓝	Olive	橄榄绿
Purple	紫 色	Yellow	黄 色
Aqua	浅蓝绿	Blue	蓝 色
Green	绿 色	Fuchsia	紫红色
White	白 色	Teal	暗蓝绿

10.1.3 HTML 页面编程简例

我们可以使用上述 4 种标签编程制作一个只显示“你好!”的页面。代码如下:

```
<HTML>
  <HEAD>
    <TITLE>Hello!</TITLE>
  </HEAD>
  <BODY>
    你好!
  </BODY>
</HTML>
```

我们可以使用 Windows 系统的“记事本”程序来输入这些代码,如图 10.1 所示,并将该 HTML 文件起名保存为“Hello.html”。

然后,我们再到 Internet Explorer 浏览器中浏览这个网页,方法是将 HTML 文件名及其路径输入到 Internet Explorer 的地址框内。比如 Hello.html 文件位于“c:\my documents\”文件夹中,则输入的路径为“c:\my documents\hello.html”,回车后浏览器即执行 Hello.html 页面文件,并立即显示“你好!”的网页信息,如图 10.2 所示,<TITLE>包含的文本“Hello”显示在网页窗口的标题栏,<BODY>包含的内容“你好!”显示在网页中。

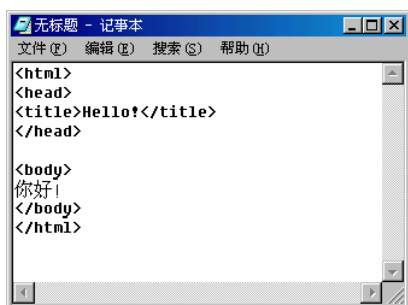


图 10.1 输入 HTML 文件

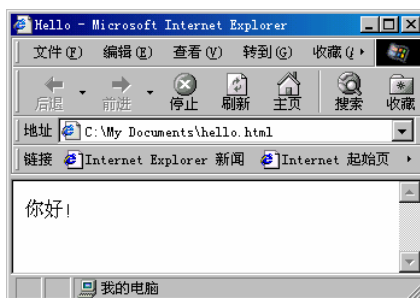


图 10.2 显示“你好！”

10.2 HTML 标签及使用

为便于大家更容易地切入 HTML 标签及其编程，我们这里再介绍一些比较常用而基本的 HTML 标签。学习这些标签对理解和掌握 HTML 的标签是大有帮助的。

10.2.1 文本类标签及其属性

HTML 文本类标签主要用于对显示文本和段落的控制，包括标题级别设置、换行设置、字符格式设置等。

行标签

“BR”即 Break，是换行标签，它是单独出现的，作用相当于插入一个回车符。如果没有换行标签的话，Web 浏览器将根据浏览器窗口的宽度尽可能长地显示文本，而不是按照 HTML 页面中的文本格式显示。

标题级标签<H_i> (i=1,2,3,4,5,6)

<H_i>标签是成对出现的，夹在<H_i>和</H_i>之间的文字是HTML页面中的标题。标题文字都用黑体显示，上级标题总比下面各级标题要更大些、粗些，具体有多大则因浏览器的不同而不同。<H₁>标签共分六级，其中<H₁>标签所括起的文字是第一级标题，最大最粗；<H₆>标签所括起的文字是最后一级标题，最小最细。通过下面的程序我们就可以观察到各级标题的效果：

```
<html>
<head>
```

```
<title>H1标签测试文档。</title>
</head>
<body>
<H1>这是第一级标题。</H1>
<H2>这是第二级标题。</H2>
<H3>这是第三级标题。</H3>
<H4>这是第四级标题。</H4>
<H5>这是第五级标题。</H5>
<H6>这是第六级标题。</H6>
</body>
</html>
```

其显示效果如图 10.3 所示。

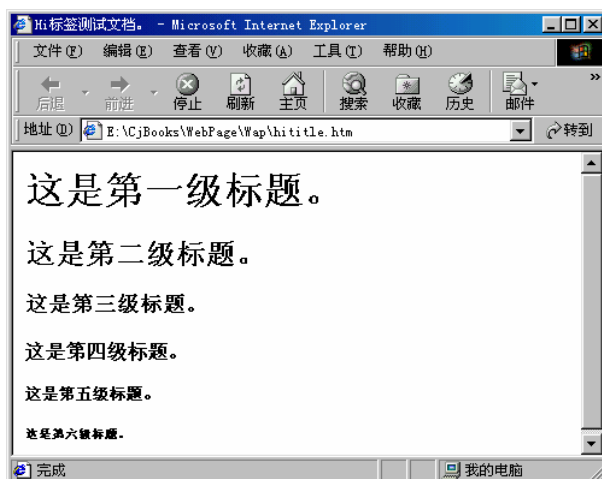


图 10.3 页面中各级标题显示效果图

<H_i>标签隐含有换行标签的作用，过长的文本可从下一行开始显示。<H_i>标签的属性有COLOR、ALIGN，分别标识标题的颜色和位置。COLOR属性的参数值取值范围和<BODY>标签的属性一样，可以是 16 种颜色(详见表 10.1)中的任意一种。ALIGN属性的参数值为left、center或right之一，分别表示标题位于浏览器窗口的左侧、中间和右侧，具体用法我们下面会介绍。

段落标签<P>

P 即 paragraph(段落)，<P>标签用于划分段落，作用是换行并插入一个空白行，它可以

单独使用，也可以成对使用。

当<P>和</P>标签成对使用时，可以添加 ALIGN 属性，用以标识段落在浏览器中的位置。同上所述，ALIGN 属性的参数值为 left、center 和 right 之一，分别表示<P>和</P>段落位于浏览器窗口的左侧、中间和右侧。通过下面的例子就可以看到这 3 种对齐效果：

```
<html>
<head>
<title>这是一个段落位置测试文档。</title>
</head>
<body>
<P ALIGN=left>段落位于左侧。</P>
<P ALIGN=center>段落位于中间。</P>
<P ALIGN=right>段落位于右侧。</P>
</body>
</html>
```

其显示效果如图 10.4 所示。



图 10.4 段落标签的位置对齐属性

水平线标签<HR>

<HR>标签是单独使用的标签。HR 即 Horizontal Rule，它的作用是换行并在该行下画一条水平直线，直线的上下两端都会留出一定的空白。<HR>标签的属性有：SIZE、WIDTH 和 ALIGN，它们的作用解释如下：

- (1) SIZE 属性。用于规定水平线的高度，该属性的参数值必须是数字。
- (2) WIDTH 属性。用于规定水平线的宽度，该属性的参数值可以是数字(代表字符数)或百分数(占浏览器宽度的百分比)。缺省时，水平线占据整个浏览器窗口宽度。



(3) **ALIGN** 属性。当水平线宽度小于浏览器窗口宽度时,使用该属性规定水平线在浏览器窗口的位罝。**ALIGN** 属性的参数值为 **left**、**center** 和 **right** 之一,分别表示该水平线位于浏览器窗口的左侧、中间和右侧,前已有例,此处不再复举。

预格式化标签<PRE>

<PRE>标签是预格式化标签,它是成对出现的。**Web** 浏览器按编辑文档时的字符位罝将<PRE>和</PRE>标签之间的内容忠实地、一成不变地显示出来。

一般说来,如果不使用预格式化标签,**Web** 浏览器在显示 **HTML** 页面时只保留格式符而忽略原页面中的回车和空格,并根据需要对原文档的字体、大小进行调整。

字符格式标签

HTML 页面的字符格式标签主要有(Bold)、<I>(Italic)和<U>(Underlined)三种,它们都是成对出现的。运行 **HTML** 页面时,和之间的内容将显示为粗体文字,<I>和</I>之间的内容将显示为斜体文字,<U>和</U>之间的内容将显示为带下划线的文字。

地址标签<ADDRESS>

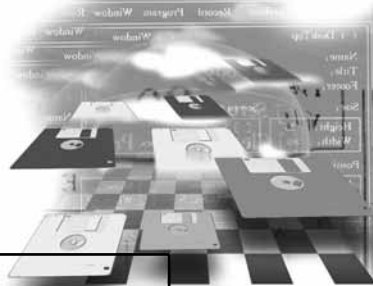
<ADDRESS>标签是地址标签,它是成对出现的,一般放在文档体的首部或尾部。<ADDRESS>和</ADDRESS>之间的内容一般是有关作者的信息,包括作者的姓名、身份、地址等等,在 **Web** 浏览器窗口中用黑体或斜体显示。

注释标签

在编写 **HTML** 页面时,为了使文档清晰易懂以及方便日后的维护,我们常常在文档中加入一些注释语句。在 **HTML** 页面中插入注释语句是通过使用注释标签<!--和-->来实现的,包含在标签<!--和-->之间的内容就是注释语句的内容。在运行时,浏览器将忽略嵌套在注释标签之间的内容,不会对其进行解释执行。因此,我们也可以用注释标签来包含暂时不用的页面内容,这在我们调试 **HTML** 页面效果时是非常有用的。

10.2.2 图像标签及其属性

图像可以使 **Web** 页面更加生动美观、富有生机。**Web** 浏览器可以显示 **JPEG** 和 **GIF** 图像。其中 **GIF** 图像最多只能使用 256 种颜色(即只能保存为 8 位颜色),而 **JPEG** 格式可保存为 24 位,对具有连续色调的图像有最佳效果。不过,**GIF** 图像虽然在图像质量上不及 **JPEG**



图像, 但其所占存储空间小, 下载速度快。因此应视不同情况而决定使用哪种格式的图像: 对于徽标、公司厂标等在每一主页显示, 要求能快速下载并能在低配置的系统中查询的图像应采用 GIF 格式, 而对于扫描图片、艺术作品这种对显示质量要求很高的图像则应采用 JPEG 格式。

在 HTML 页面中插入图像是通过标签来实现的, 该标签共有 9 个属性, 除属性 SRC 是不可缺省的外, 其它属性都可选的。

(1) SRC 属性。SRC 即 Source, 该属性用于指出被引用的图像文件所在位置。SRC 属性的参数值就是图像文件的 URL, URL 的表示方法有绝对表示法和相对表示法两种。

绝对表示法范例:

相对表示法范例:

只有当图像文件和 HTML 页面在同一目录下时才能采用相对表示法来标识图像文件的位置。

(2) WIDTH 和 HEIGHT 属性。标签用 WIDTH 和 HEIGHT 这两个属性来规定图像的大小。其中 WIDTH 属性用于确定图像的宽度, HEIGHT 属性用于确定图像的高度。这两个属性的参数值都是数值, 表示图像宽度(高度)所占的像素点数。缺省时, Web 原始图像文件的大小和浏览器窗口的大小自动调整图像的显示尺寸。

(3) ALIGN 属性。它的参数值为 top、middle 或 bottom, 分别表示与图像相邻的文字位于图像的左上方、左面中间和右下方。

(4) ALT 属性。用户在浏览网页时, 常常会为了节省时间或其他原因而选择不下载图片, 加入 ALT 属性可以在原先显示图片的地方显示一些有关图像的信息。ALT 属性的参数值就是网页编辑者希望在原来显示图像处出现的文字信息, 一般是图像的名称或简要说明。例如:

```
<IMG src="cake.jpg" ALIGN=bottom ALT="A PICTURE OF CAKE">  
Happy Birthday to you!(JPEG 格式)  

```

(5) VSPACE 和 HSPACE 属性。标签用 VSPACE 和 HSPACE 这两个属性来确定图像与其相邻对象之间的空白大小。VSPACE 属性用于指定图像与其垂直方向上相邻对象



之间的距离，HSPACE 用于指定图像与其水平方向上相邻对象之间的距离。这两个属性的参数值都是数值，表示空出距离的像素点数。

(6) BORDER 属性。Web 浏览器在调用图像时会根据浏览器窗口和原始图像大小的不同给图像加上不同宽度的边框。网页编辑者可以利用 BORDER 属性来指定图像边框的宽度或取消边框(BORDER=0)。BORDER 属性的参数值也是数值，表示边框宽度所占的像素点数。下面的语句就可以为显示的图像 cake.jpg 加上宽度为 15 个像素的边框。

```
<IMG src="cake.jpg" ALIGN=bottom border=15 >  
Happy Birthday to you!(JPEG 格式)
```

(7) ISMAP 属性。这一属性与文档的超链接有关，我们把它放到后面讲解。

10.2.3 列表类标签及其属性

HTML 页面中的列表主要有 5 种，即无序列表、排序列表、目录列表、菜单列表和描述性列表，下面我们分别介绍一下。

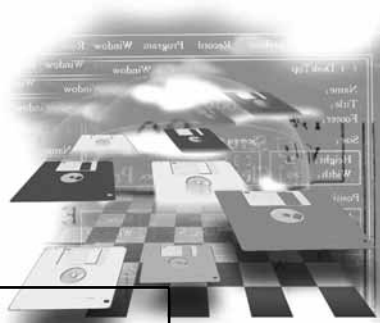
无序列表

在 HTML 页面中插入无序列表是通过标签和标签来实现的。“UL”即 Unordered List，指无序列表标签。标签是成对标签，首标签和尾标签之间的内容就是无序列表的内容。无序列表中的每一列表项开头必须用标签进行标识。

“LI”即 Listed Item，指项目标签。标签是单独出现标签，标签后的文字为列表项的内容，直至碰到下一个项目标签或列表的尾标签。Web 浏览器在显示无序列表时，会在每一列表项的开头加上无序列表的标签符号。

排序列表

在 HTML 页面中插入排序列表是通过和标签来实现的。“OI”即 Ordered List，是指排序列表标签。标签是成对出现的，首标签和尾标签之间的内容就是排序列表的内容。排序列表中的每一列表项也必须用标签进行标识。Web 浏览器在显示排序列表时，会自动对列表进行排序，并在每一列表项开头加上以阿拉伯数字形式表示的序号。我们可以在标签中添加 start 属性来指定序列的起始号。缺省时，序列的起始号为 1。



目录列表

HTML 使用<DIR>和标签来定义目录列表。“DIR”即 Directory，指目录列表标签。<DIR>标签是成对出现的，首标签<DIR>和尾标签</DIR>之间的内容就是目录列表的内容。目录列表的列表项开头也必须用项目标签标识。一般说来，目录列表项长度较短，典型的目录列表每项长度不超过 20 个字符。

菜单列表

HTML 使用<MENU>和标签来定义菜单列表。菜单列表标签<MENU>是成对标签，首标签<MENU>和尾标签</MENU>之间的内容就是菜单列表的内容。菜单列表每一列表项开头也必须用项目标签进行标识。一般来说，菜单列表每项的长度不超过一行。

描述性列表

HTML 使用<DL>、<DT>和<DD>标签来定义描述性列表。“DL”即 Definition List，指描述性列表标签。<DL>标签是成对标签，首标签<DL>和尾标签</DL>之间的内容就是描述性列表的内容，由一系列用描述项标签<DT>或解释项标签<DD>标识的列表项组成。<DT>和<DD>标签都是单独标签，位于列表项的开头，分别表示该项是描述项和解释项。和标签一样，<DT>和<DD>标签也隐含有
标签的作用，Web 浏览器在显示时会自动换行。

<DT>和<DD>标签通常是成对出现的，即一个描述项对应于一个解释项。一个描述项也可以对应于几个解释项，但最好不要出现几个描述项对应于同一个解释项的情况。

HTML 中的列表是可以嵌套的，也就是说，一个列表的列表项可以是另一个相同或不同类型的列表。

10.2.4 表格类标签及其属性

表格是组织 HTML 页面内容的有效形式，HTML 提供了与表格生成和控制有关的多个标签，我们下面就简要介绍一下。

表格标签<table>

<table>标签是表格的标识符，用以界定表格的范围。<table>标签是成对出现标签，首标签<table>和尾标签</table>之间的内容就是表格的内容。<table>标签的属性主要有 border、width、height、align、cellspacing 和 cellpadding，它们都是可选的。

(1) border 属性。border 属性的参数值是数字,表示表格边框宽度所占的像素点数; border 属性也可以不带参数值使用,仅表示该表格是有边框的。例如, <table border=10>表示该表格的边框宽度为 10 个像素点。

(2) width 和 height 属性。width 和 height 属性的作用是指定表格的大小。其中 width 属性用以规定表格的宽度, height 属性用以规定表格的高度。这两个属性的参数值可以是数字或百分数,其中数字表示表格宽度(高度)所占的像素点数,百分数表示表格的宽度(高度)占浏览器窗口宽度(高度)的百分比。例如, <table width=200 height=50%>表示该表格的宽度为 200 个像素点,高度为浏览器窗口高度的 50%。

(3) align 属性。align 属性的参数值为 left、center 和 right 之一,分别表示表格位于其相邻文字的左侧、表格水平居中和表格位于与其相邻的文字右侧。

(4) cellpadding 属性。cellpadding 属性用以指定表格各单元格之间的空隙。该属性的参数值为数字,表示单元格间隙所占的像素点数。

(5) cellspacing 属性。cellspacing 属性用于指定单元格内容与单元格边界之间空白距离的大小。该属性的参数值也是数字,表示单元格内容与上下边界之间空白距离的高度所占的像素点数以及单元格内容与左右边界之间空白距离的宽度所占的像素点数。

和普通表格一样, HTML 中的表格也是由表格标题、表头元素和具体数据元素组成的。HTML 是通过在表格标签中嵌套使用有关标签来定义这些元素的。嵌套在表格标签中使用的标签主要有<CAPTION>、<TR>、<TH>和<TD>。

表格标题标签<CAPTION>

表格标题标签<CAPTION>是成对出现标签,首标签<CAPTION>和尾标签</CAPTION>之间的内容就是表格的标题。<CAPTION>有两个属性: ALIGN 和 VALIGN,它们都是可选的。

ALIGN 属性的参数值为 left、center 和 right 之一,分别表示表格标题与表格的左沿对齐、位于表格中间和与表格的右沿对齐。缺省时,表格标题位于表格中间。

VALIGN 属性的参数值为 top 和 bottom 之一,分别表示表格标题位于表格的上方和下方。缺省时,表格标题位于表格的上方。



行标签<TR>

HTML 中的表格是按行组织的。一个表格由几行组成，就要有几个行标签与之相对应。行标签<TR>是成对标签，它必须与<TH>或<TD>标签配套使用，后者在使用时嵌套在行标签的首标签<TR>和尾标签</TR>之间，用于规定该行中各单元格的内容。<TR>标签出有 2 个属性：ALIGN 和 VALIGN，它们都是可选的。

ALIGN 属性的参数值为 left、center 和 right 之一，分别表示该行中各单元格内容是左对齐、水平居中和右对齐的。ALIGN 属性的缺省值为 left。

VALIGN 属性的参数值为 top、middle 和 bottom 之一，分别表示该行中各单元格内容是紧靠上沿、垂直居中和紧靠下沿的。VALIGN 属性的缺省值为 middle。

单元格标签

<TH>和<TD>标签都是用于规定单元格内容的。表头标签<TH>用于规定表头元素的内容，表头元素一般位于每列的首行，用以说明该列的具体数据是关于哪个对象的。<TH>标签是成对出现标签，首标签<TH>和尾标签</TH>之间的内容就是位于该单元格中的表头元素内容。数据标签<TD>也是成对出现标签，首标签<TD>和尾标签</TD>之间的内容就是该单元格中的具体数据。除了表头元素是以黑体显示这一点外，表头元素和具体数据元素几乎没有什么区别。由于都是用于规定单元内容的，所以<TH>和<TD>标签的所有属性及相应的属性功能是完全一样的。<TH>和<TD>标签的属性有 rowspan、colspan、align 和 valign，它们都是可选的。

(1) rowspan 属性。rowspan 属性的参数值是数值，表示该单元格所跨的行数。该属性的缺省值为 1。

(2) colspan 属性。colspan 属性的参数值是数值，表示该单元格所跨的列数。Colspan 属性的缺省值也是 1。

(3) align 属性。align 属性用于规定单元格内容在水平方向上的位置。属性的参数值为 left、center 和 right 之一，分别表示该单元格内容是左对齐、水平居中和右对齐的。缺省时，单元格内容是左对齐的。

(4) valign 属性。valign 属性用于规定单元格内容在垂直方向上的位置。属性的参数值为 top、middle 和 bottom 之一，分别表示该单元格内容是紧靠上沿、垂直居中和紧靠下沿的。缺省时，单元格内容是垂直居中的。

10.2.5 文档超链接标签

HTML 最显著的优点就在于它支持文档的超链接，用户可以很方便地在不同文档以及同一文档的各段段落之间跳转。

HTML 中的链接包括两部分：锚标和目标点。锚标就是链接的源点，当鼠标移到锚标处时会变成小手状。此时，用户通过点击鼠标就可以到达链接的目标点。

HTML 是通过链接标签来实现超链接的。链接标签<A>是成对标签，首标签<A>和尾标签之间的内容就是锚标。<A>标签有一个不可缺省的属性 href，用于指定链接目标点的位置。HTML 支持的超链接主要有 3 种：不同文档之间的跳转、跳转到标记位置、链接地图。

跳转到其他文档

用户可以通过点击锚标从当前文档直接跳转至目标文档。如前所述，首标签<A>和尾标签之间的内容就是锚标，一般是对目标文档的简要说明。此时，href 属性的参数值是目标文档的 URL。程序举例如下：

```
<html>
<head>
<title>文档链接范例程序</title>
</head>
<body>
<P><HR></P>
<A href= "apple.html" >有关苹果的说明</A>
<P><HR></P>
下面是相应的图像锚标：
<A href= "apple.html" ><IMG SRC= "apple.gif" ></A>
<HR>
</body>
</html>
```

如果用户点击文字锚标“有关苹果的说明”或点击苹果图像，浏览器就会载入目标文档 apple.html。锚标文字和普通文字在外观上有很大的不同，一般来说，它被显示为带下划



线的亮蓝色文字。

跳转到标记位置

浏览器程序在载入目标文档后将自动从文档的开头部分开始显示。然而，用户需要的部分往往并不在文档的开头。如果不加以记号标记，用户将花费大量的时间和精力在文档(特别是一些长文档)中进行查找。为了方便用户，我们可以利用<A>标签的 NAME 属性在目标位置添加记号，然后在锚标中进行引用。

HTML 中标识记号的格式为：

 目标点

引用记号的格式为：

 锚标内容

链接地图

链接地图是一种很形象的超链接方式。它将链接源点的图像划分为若干个区域，每个区域都对应着自己的目标文档。链接地图的语法与其他两种超链接方式有很大的不同，我们来看一个例子：

```
<html>
<head>
<title> An Example of linkmap </title>
<body>
<center>
<p>
<IMG SRC="map.jpg" ISMAP USEMAP="#ss">
<map NAME="ss">
<area shape="rect" coords="100,76 162,123" href="畜牧场.jpg">
<area shape="rect" coords="276,80 339,124" href="田园风光.jpg">
<area shape="rect" coords="104,200 163,244" href="sunset.html">
<area shape="rect" coords="262,195 351,240" href="采摘果实.jpg">
<area shape="circle" coords="223,169 253,169" href="m1.jpg">
</map>
</body>
</html>
```

在定义链接地图时，首先要为链接源点的图像添加 ISMAP 属性，将图像定义为地图。



然后必须使用 USEMAP 属性说明该图像是遵循哪个地图定义的。USEMAP 属性的参数值为“#地图名”。

HTML 是通过<map>标签和<area>标签来定义地图的。地图标签<map>有一个不可缺省的属性 NAME，它的参数值是地图的名字。区域的划分是通过在<map>标签中嵌套使用<area>标签来实现的。<area>标签有 3 个属性：shape、coords 和 href，分别用以规定该区域的形状、范围和对应的目标文档。

在本例中<area shape="rect" coords="100,76 162,123" href="畜牧场.jpg">语句定义了一个矩形区域，矩形的左上角座标为(100, 76)，右下角座标为(162, 123)。在该区域内任何位置单击鼠标都可以链接到目标文档“畜牧场.jpg”。<area shape="circle" coords="223,169 253,169" href="m1.jpg">语句定义了一个圆形区域，(223, 169)为圆心座标，(253, 169)是圆上任意一点的座标。当用户将鼠标移到定义的链接区域内时，鼠标就会变成手的形状。此时，用户点击鼠标就可以链接到相应的目标文档。

10.2.6 表单类标签与交互界面

Internet 用户不仅希望能从 Web 页面中获得有用的信息，而且可能希望把自己的信息通过 Web 页面反馈给 Web 服务器端的业务提供商。实现这一信息反馈功能的就是表单和交互界面，HTML 专门为此提供了表单标签和实现交互功能的标签。下面我们就一一介绍这些标签。

表单标签<form>

表单标签<form>是成对出现的标签，首标签<form>和尾标签</form>之间的内容就是一个表单。<form>标签有 2 个不可缺省的属性：ACTION、METHOD，和一个可选的属性：NAME。

(1) ACTION 属性。大家知道，WWW 是采用客户机/服务器工作方式的。用户在浏览器端通过 HTML 表单提交的数据将被传送到 Web 服务器中，由相应的处理程序进行处理。ACTION 属性的作用就是指出该表单所对应的处理程序的位置。它的参数值就是该程序的 URL。

(2) METHOD 属性。METHOD 属性用于指定该表单的运行方式。属性的参数值为 get 和 post 之一。METHOD=get 表示该表单主要是从服务器中获取信息，它传送给服务器的反



馈信息长度不能超过 255 个字符；METHOD=post 表示该表单主要是向服务器发送信息的，它传送给服务器的反馈信息长度没有限制。

例如，<form ACTION= “http://www.online.sh.cn/asp/query.asp METHOD=post>表示该表单的反馈信息将被传送到相应的处理程序——上海热线服务器 asp 目录下的 query.asp 程序中去。该表单的作用主要是向服务器发送信息。

(3) NAME 属性。除了上面介绍的 2 个不可缺省的属性外，<Form>标签还有一个可以缺省的属性 NAME，该属性用以指定表单的名称。

输入标签<input>

输入标签<input>是单独标签，它必须嵌套在表单标签中使用，用于定义一个用户输入项。<input>标签有 2 个固定属性：NAME 和 TYPE。其中 NAME 属性的参数值是相应处理程序中的变量名，Web 服务器将把这条输入信息的值赋予 NAME 属性规定的变量。TYPE 属性用于指定该输入项提供的输入方式。在不同的输入方式下，<input>标签的格式略有不同。TYPE 属性的参数值可以是以下 8 项之一：

(1) text(文本框)。表示该输入项的输入信息是字符串。此时，浏览器会在相应位置显示一个文本输入框，供用户输入信息。当 TYPE=text 时，<input>标签除了有两个不可缺省的属性 NAME 和 TYPE 外，还有三个可选的属性：VALUE、SIZE 和 MAXLENGTH。

VALUE 属性用于指定文本输入框的初始值，属性的参数值就是浏览器最初显示在文本框中的内容。SIZE 属性用于指定文本框的长度，属性的参数值为数字，表示该文本输入框所能显示的最大字符数。MAXLENGTH 的参数值也是数字，表示该文本输入框允许用户输入的最大字符数。MAXLENGTH 属性的参数值总是大于或等于 SIZE 属性的参数值，当输入的字符数超过文本输入框的长度时，用户可以通过移动光标来查看超过部分的内容。

(2) password(密码)。表示该输入项的输入信息是密码。TYPE=password 方式下的输入和 TYPE=text 方式下的输入基本相同，但是浏览器并不在文本输入框中显示用户输入的密码，而是用 “*” 号来代替每个密码字符，以保证密码的安全。程序举例如下：

```
<html>
<head>
<title> An example of form </title>
</head>
<body>
```

```
<P><HR></P>
```

欢迎到 小书虫网上书店购书, 请键入您的用户名及密码:


```
<Form Action="http://www.shop.com/asp/pass.asp" METHOD="post "
Name="form1">
```

```
<p> Username: <Input Type=text SIZE=16 NAME="user">
```

```
<BR><p>
```

```
password: <Input Type=password SIZE=16 Name="pw"><BR>
```

```
<HR>
```

```
</Form>
```

```
</body>
```

```
</html>
```

显示效果如图 10.5 所示。

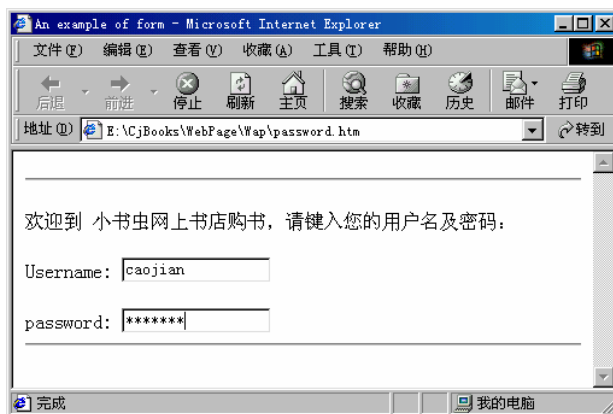


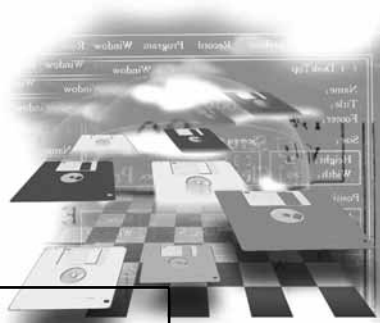
图 10.5. 文本和密码输入显示效果图

(3) checkbox(复选项)。表示该输入项是一个复选项, 用户可同时选中表单中的一个或几个复选项作为输入信息。当 TYPE=checkbox 时, 除了 TYPE 和 NAME 属性外, <input> 标签还有两个属性: VALUE 和 CHECKED。

VALUE 属性的参数值就是当该项被选中并提交后, Web 浏览器要传送给服务器的数据。因此, VALUE 属性的参数值必须与选项内容相同或基本相同, 该属性是不可缺省的。

CHECKED 属性用于指定复选项的初始状态表示该复选项在初始时是被选中的。

(4) radio(单选项)。表示该输入项是一个单选项。单选项是互斥的, 用户只能选中表单中所有单选项中的一项作为输入信息。因此, 所有单选项 NAME 属性的参数值是相同的。当 TYPE=radio 时, <input> 标签还有两个属性: VALUE 和 CHECKED, 它们的使用方法和



功能与 `TYPE=checkbox` 时完全一样，这里不再赘述。

(5) `hidden`(隐藏项)。`TYPE=hidden` 表示输入项将不在浏览器窗口中显示。如果用户不想显示某些选项而又不愿将它们从文档中删去的话，那么就可以通过将这些选项中 `TYPE` 属性的参数值改为 `hidden` 来达到上述目的。另外，该元素还经常用于判断用户是从哪个界面登录网站的。

(6) `submit`(提交按钮)。当 `TYPE=submit` 时，浏览器会在相应位置产生一个提交按钮。当用户单击该按钮时，浏览器就会将表单的输入信息传送给服务器。提交按钮的 `NAME` 属性是可以缺省的。除 `NAME` 属性外，它还有一个可选的属性 `VALUE`，用于指定显示在提交按钮上的文字。`VALUE` 属性的缺省值是由用户使用的浏览器确定的：在 Netscape 中为“Submit query”，IE 中为“提交查询内容”。在一个表单中必须有提交按钮，不然将无法向服务器传送信息。

(7) `image`(图像按钮)。当 `TYPE=image` 时，浏览器会在相应位置产生一个图像按钮。当用户单击该按钮时，浏览器就会将表单的输入信息传送给服务器。在使用图像按钮时，必须在 `<input>` 标签中添加 `SRC` 属性指出图像所在位置。另外，很多在图像标签中使用的属性规定也可以在图像按钮中使用。

(8) `reset`(还原按钮)。当 `TYPE=reset` 时，浏览器会在相应位置产生一个还原按钮。当用户单击该按钮时，浏览器就会清除表单中所有的输入信息而恢复到初始状态。和提交按钮一样，还原按钮也有两个可以缺省的属性：`NAME` 和 `VALUE`。其中 `NAME` 属性的值是该还原按钮的名称，`VALUE` 属性则用于指定显示在还原按钮的文字。`VALUE` 属性的缺省值由用户使用的浏览器决定：Netscape 中为“Reset”，IE 中为“还原”。

用于定义列表框的 `<select>` 标签和 `<option>` 标签

HTML 是通过 `<select>` 和 `<option>` 标签来定义输入列表框的。列表框标签 `<select>` 是成对出现的标签，首标签 `<select>` 和尾标签 `</select>` 之间的内容就是一个列表框的内容。`<select>` 标签必须与 `<option>` 标签配套使用，后者用于定义列表框中的各个选项。`<select>` 标签有两个属性：`NAME` 和 `SIZE`，其中 `NAME` 属性是不可缺省的。

(1) `NAME` 属性。用于指定输入列表框的名字。

(2) `SIZE` 属性。`SIZE` 属性是可选的，用于定义列表框的长度。`SIZE` 属性的参数值是数字，表示显示在列表框中的选项数目。当 `SIZE` 属性的参数值小于列表框中的列表项数目时，浏览器会为该列表框添加滚动条，用户可以使用滚动条来查看所有的选项。`SIZE` 属性的缺

省值为 1。

HTML 使用<option>标签来定义列表框中的选项。<option>标签是单独标签，它必须嵌套在列表框标签中使用，一个列表框中有几个选项，就要有几个<option>标签与之相对应。<option>标签有两个属性：VALUE 和 SELECTED，它们都是可选的。其中 VALUE 属性的参数值是当该项被选中并提交后，Web 浏览器传送给服务器的数据。缺省时，浏览器将传送选项的内容。CHECKED 属性用来指定选项的初始状态，表示该选项在初始时是被选中的。例如，通过下面的简单程序就可以观察该属性的效果：

```
<html>
<head>
<title>An example of SELECT</title>
<HR>
<form      ACTOPN="http://www.shop.com/asp      /item.asp"      METHOD="post "
Name="form1">
  <P>请选择您想去的商店(食品类): <BR>
  <P><select Name="store" size=3>
    <option selected>欣欣百货商店
    <option VALUE= "富兴百货">富兴百货商店
    <option>联华超市
    <option>麦德隆超市
    <option>第一百货商店
    <option>第二百货商店
  </select>
  <P>请在选择完毕后单击确认按钮:
  <Input type=submit value="确 认">
</form>
<HR>
</body>
</html>
```

这段程序的运行效果如图 10.6 所示。

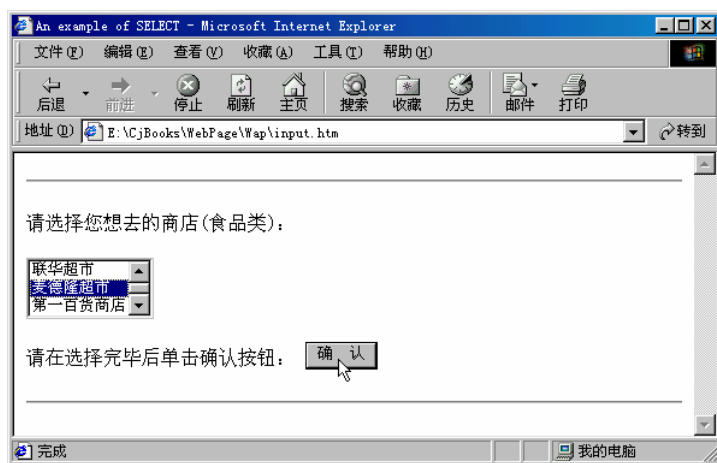


图 10.6 列表输入框显示效果图

文本区域标签<textarea>

HTML 用<textarea>标签来定义高度超过一行的文本输入框。<textarea>标签是成对出现标签，首标签<textarea>和尾标签</textarea>之间的内容就是显示在文本输入框中的初始信息。<textarea>标签有 3 个属性：NAME、ROWS 和 COLS。

(1) NAME 属性。用于指定文本输入框的名字。

(2) COLS 属性。COLS 属性用于规定文本输入框的宽度。属性的参数值是数字，表示一行所能显示的最大字符数。如果输入信息中有一行或几行的字符数大于 COLS 属性的参数值，浏览器会为该文本输入框添加水平滚动条。

(3) ROWS 属性。ROWS 属性用于规定文本输入框的高度。属性的参数值是数字，表示该文本输入框所占的行数。浏览器会自动为高度超过一行的文本输入框添加垂直滚动条。但是当输入信息的行数小于或等于 ROWS 属性的参数值时，滚动条将不起作用。

表单元素的分组

HTML 允许网页编辑者对表单元素进行分组。通过对表单元素进行分组，我们可以将实现同一功能的表单元素放在一个集合里，并可以对同一集合中的表单元素进行统一设置。这样会。使页面结构更加清晰。

HTML 中表单元素的分组是通过表单集合标签<FIELDSET>和信息描述标签<LEGEND>来实现的。表单集合标签<FIELDSET>是成对出现标签，它用于指定一个表单集合的范围。首标签<FIELDSET>和尾标签</FIELDSET>之间的内容就是该表单集合的所有内容，包括所有表单元素以及描述信息。信息描述标签<LEGEND>是嵌套在表单集合标签

<FIELDSET>和</FIELDSET>之间使用的。它也是成对出现标签，首标签<LEGEND>和尾标签</LEGEND>之间的内容通常是对表单集合的描述。描述信息在浏览器窗口中显示位置是由<LEGEND>标签的 ALIGN 属性来规定的。ALIGN 属性的属性值可以是 left、right 或 center 之一，分别表示描述信息将显示在表单集合区域的左上角、右上角和区域上方中央。

10.2.7 框架类标签及其属性

框架网页是一种特殊的 HTML 网页，可用将浏览器窗口分割成不同的小窗口区域，显示若干个 Web 页面。页面中的框架内可以显示不同的 HTML 页面，能够独立翻滚浏览。HTML 使用<Frameset>、<Frame>与<noFrames>标签来定义和控制框架，下面我们就分别介绍一下。

框架设置标签<Frameset>

框架设置标签<Frameset>标签是成对出现标签，首标签<Frameset>和尾标签</Frameset>之间的内容就是使用框架的 HTML 页面主体部分。在使用框架的 HTML 页面中不能出现<body>标签，否则会导致 Web 浏览器忽略所有的框定义而只显示<body>和</body>之间的内容。

<Frameset>标签的作用是将窗口分割为若干个子窗口，子窗口的数目取决于嵌套在该标签中的<Frame>标签的数目。<Frameset>标签有两个属性：Rows 和 Cols，分别用来确定各子窗口的高度和宽度，格式为：

<Frameset Rows= “值 1,值 2,...,值 n” >

<Frameset Cols= “值 1,值 2,...,值 n” >

各参数值之间用逗号分隔，依次表示各个子窗口的高度(宽度)。这两个属性的参数值可以是数字、百分数或符号“*”。其中，数字表示子窗口高度(宽度)所占的像素点数；百分数表示子窗口高度(宽度)占整个浏览器窗口高度(宽度)的百分比；符号*如果只出现一次，即其他子窗口的大小都有明确定义时，表示该子窗口的大小将根据浏览器窗口的大小而自动调整。当符号*出现一次以上时，表示按比例分割浏览器窗口的剩余空间。

例如，<Frameset Cols= "40%, 2*, * " >表示将浏览器窗口分割为 3 列，第一个子窗口在第一列，窗口宽度为整个浏览器窗口宽度的 40%；第二个子窗口在第二列，占浏览器窗口剩余空间的 2/3，即其宽度为整个浏览器窗口宽度的 40%；第三个子窗口占剩余空间的 1/3，



宽度为整个浏览器窗口宽度的 20%。

标识框架子窗口的标签<frame>

HTML 用<frame>标签来标识子窗口。<frame>标签是嵌套在框架设置标签<Frameset>和</Frameset>中使用的单独标签。在<Frameset>中定义了多少个子窗口，就应该有多少个<frame>标签与之匹配，依次定义第 1,2,...,n 个子窗口的性质。<frame>标签共有 7 个属性，除 SRC 属性是不可缺省的以外，其他属性都是可选的。

(1) SRC 属性。SRC 属性的参数值是显示在该子窗口中的 HTML 页面的 URL。

(2) NAME 属性。用于定义子窗口的名称。

(3) frameborder 属性。属性的参数值为 1 或 0。当参数值为 1 时，表示该子窗口有边框；为 0 时表示该子窗口没有边框。Frameborder 属性缺省值为 1。

(4) bordercolor 属性。用以规定子窗口的边框颜色，如果在一个以上的<frame>标签中定义了子窗口的边框颜色，则以第一次指定的颜色为准。在指定边框颜色时，可以使用颜色的 RGB 代码或直接使用该颜色对应的英文单词。Bordercolor 属性的参数值可以是表 10.1 所示 16 种颜色中的任意一种。

(5) scrolling 属性。属性的参数值为 yes、no 和 auto 之一。参数值为 yes 时表示该子窗口始终有滚动条；参数值为 no 时表示该子窗口始终没有滚动条；参数值为 auto 时表示当文档内容超出窗口范围时，浏览器将自动为该子窗口添加滚动条。Scrolling 属性的缺省值为 auto。

(6) marginwidth 和 marginheight 属性。这两个属性的用于指定显示内容与窗口边界之间的空白距离大小，其中 marginwidth 用于确定显示内容与左右边界之间的距离；marginheight 用于确定显示内容与上下边界之间的距离。这两个属性的参数值都是数字，分别表示左右边距所占的像素点数。

不支持框架的提示标签<noframes>

使用<noframes>标签可以在用户浏览器不支持框架显示时告知用户一些有关信息，以免用户对空白窗口画面感到莫名其妙。<noframes>标签是成对出现标签，首标签<noframes>和尾标签</noframes>之间的内容就是网页编辑者希望告诉用户的信息。如：“您使用的浏览器不支持框架显示”。虽然常用的两种 Web 浏览器——IE 和 Netscape 都支持框架显示，但是为了加强文档的适用性，使我们编写的 HTML 页面能适应各种类型的 Web 浏览器，还是应

该养成使用这个标签的习惯。

HTML 框架的基本结构

使用框架的 HTML 页面基本结构如下：

```
<html>
<head>
<title>文档标题</title>
</head>
<!-- 以下标签将浏览器窗口分割为 n 个子窗口-->
<Frameset  Cols= "值 1, 值 2, ..., 值 n " >
<frame  SRC= "1.html" >
<frame  SRC= "2.html" >
.
.
.
<frame  SRC= "n.html" >
</Frameset>
<noframes>您使用的浏览器不能显示框架</noframes>
</html>
```

<Frameset>标签是可以嵌套使用的，也就是说，可以将其中某一个或某几个子窗口再划分为若干个更小的窗口。

10.3 HDML 语言编程基础

学习了 HTML 语言的编程方法和常用标签后，再来学习 WAP 的 HDML 编程就容易多了。与 HTML 网页类似，使用 HDML 语言编写的 WAP 网页通常亦称为 HDML 网页、HDML 页面或 HDML 文件。目前产生 HDML 文件的方法主要是手工编制，我们这里也是针对这种编制方法进行讲解。



10.3.1 HDML 语言的开发环境

与 WML 语言类似, HDML 语言本质上也是用 XML 1.0 来定义的。目前能够比较好地提供 HDML 开发环境的软件包是 Phone.com 公司推出的 UP.SDK, 它同时包括了一个 UP.Phone 手机浏览器的模拟器, 可以像真的手机一样上网, 实现 HDML 页面的测试。UP.SDK 同时提供了 Perl、C 及 COM 的库和库函数, 可以方便而功能强大地实现 HDML 与 WML/WMLScript 的编程。我们的随书光盘中提供了这个软件包, 大家可以安装 UP.SDK, 建立 HDML 的开发环境, 实际测试一下编写的 HDML 页面。

10.3.2 HDML 页面

HDML 和 HTML 在语法上有许多相似的地方, 它也采用“<标签名 属性=值>”的标签形式来格式化页面。HDML 的标签也是不区分大小写的, 但 HDML 中涉及的变量是区分大小写的, 这一点与 WML/WMLScript 的变量命名规则相同。

HDML 页面也包含两种信息, 其一是页面本身的内容, 如文本、图形等; 其二是表示页面元素、格式、超链接等的 HDML 标签(也称为标记)。编写好的 HDML 页面文件可以被 WAP 微浏览器执行, 浏览器执行 HDML 文件遇到 HDML 标签时, 就会将页面的内容按照标签定义的格式显示出来。

HDML 的标签也使用小于号(<)和大于号(>)括起来, 即采用“<标签名>”的形式。标签也分单独出现的标签和成对出现的标签两种, 成对出现的标签都由首标签和尾标签组成, 它们又分别称为起始标签和终止标签。首标签的格式为“<标签名>”, 尾标签的格式为“</标签名>”。

使用 HDML 编写的同一 WAP 页面, 当由不同类型(如不同品牌的手机)的浏览器解释时, 也可能发生格式结果不一样的情况, 也就是说, 同一个 HDML 标签在不同浏览器中显示的结果可能不一样。因此编写 WAP 网页时需要注意这些问题, 要了解浏览器的版本和无线设备的要求, 并据此写出满足要求的 HDML 页面。

HDML 页面文件也是纯文本文件, 可以用普通的文本编辑器进行编辑, 只不过保存时它的扩展名必须保存成 .hdml。我们一般采用纯文本文档编辑器, 如 Windows 中的记事本或写字板, DOS 中的文本编辑器 Edit 等来编辑 HDML 页面, 也可以使用各厂商提供的 WAP



开发工具软件包来编辑 HDML 页面。

10.3.3 HDML 页面文件的结构

我们通过一个简单的实例，说明 HDML 页面文件的基本结构。例程如下：

```
Content-type: text/x-hdml; charset=GB2312
<HDML VERSION="4.0">
  <DISPLAY name="card1">
    <center>This is a test.<br>1234567890<tab>1234567890<line>xxxxx
  </DISPLAY>
</HDML>
```

程序解释如下：

(1) 第 1 行的 Content-type 一句用于指定当前 HDML 页面采用的字符集，类似于普通 Web 页面内的<meta http-equiv="Content-Type" content="text/html; charset=gb2312">，不同的是 HDML 页面中这一行需要写在程序的开头部分。

(2) 第 2 行<HDML>标签类似于 HTML 的<HTML>标签，它不仅说明该文件是 HDML 文件，而且重要的是指明成对的<HDML>和</HDML>标签之间是浏览器与 Web 服务器连接一次所传递的信息，相当于普通浏览器的页面，在 WAP 页面中称为卡片组(DECK)，其组成单位是一个一个的卡片(CARD)。

这里的<HDML>标签还含有 1 个属性，即 VERSION，它用于指明当前所用 HDML 语言的版本，以供浏览器解释时参考。VERSION="4.0"表明这里采用的 HDML 是 4.0 版本的。

HDML 各标签属性的值需要使用双引号(" ")或单引号(' ')括起来。如果属性值里包含保留字符或特殊字符则需要使用转义序列来表示。HDML 的标签虽然不多，但属性很多，所以开发时应根据需要，选择使用恰当的属性值。

(3) 第 3 行<DISPLAY>标签可以看作 HTML 里的<BODY>，它用于指明 HDML 页面的可视内容。和 HTML 不同的是，WAP 页面的一个卡片组(DECK)可以包含多个卡片(CARD)，所以 HDML 规定每个卡片用成对的<DISPLAY>和</DISPLAY>标签包括起来，而 HTML 只有一个<BODY>即一个页面。这种规定是因为充分考虑到手机等无线设备的浏览器屏幕比较小，一次不可能显示太多的信息。这样，WAP 服务器与浏览器之间的一次连接所传递的



数据虽然是一个 HDML 文件，但它实际是一个卡片组，包含有多个页，所以仍然可以向用户展示所需的、足够多的信息。

HDML 卡片的类型有 3 种：纯文本及图形显示、选单(相当于 HTML 的 FORM)、不显示内容的 action 动作。

这里的<DISPLAY>标签还含有 1 个参数，即 name，它用于指明当前 HDML 页面即卡片的名称为“card1”。

(4) 第 4 行是当前卡片显示的内容，如果一屏显示不下，浏览器屏幕上会出现提示。具体提示符号因手机类型的不同而不同，可能是大于号(>)，也可能是竖线(|)。

10.4 HDML 标签及编程

HDML 语言为 HDML 页面的文本显示、图像显示、选单生成、行为执行和 CGI 处理等提供了多种标签，下面我们就讲解这些标签的使用方法。

10.4.1 文本标签及规定

HDML 用于显示文本的标签就是前面介绍的<DISPLAY>标签，它是一个成对出现的标签，包含的内容就作为 HDML 的卡片显示在用户浏览器屏幕上。它有 1 个 name 属性，用于指定卡片的名字。

另外，HDML 为控制文本显示还提供了一些标签，它们的名称及作用介绍如下：

(1) <TAB>标签。类似于一般意义上的 TAB 键效果，即使用该标签可以使它后面的文本移动一个 TAB 键位置后再开始显示。

(2)
标签。这是一个换行标签，与 HTML 的
标签相同，具体可参见前面(10.2.1 节)的介绍，这里不再赘述。

(3) <LINE>标签。这也是一个换行标签，但与
标签不同。当显示的一行文本太长时，<LINE>标签可使文本在屏幕上自动折行显示，而新的一行将忽略文本中的空字符，仅显示实际的非空字符。

(4) <WRAP>标签。这也是一个换行标签，与<LINE>标签有区别。显示一行较长文本时，它使文本在屏幕上自动折行的同时，还显示文本中的所有空字符。

(5) <RIGHT>标签。它可使当前行文本显示时向页面的右边界对齐。



(6) <CENTER>标签。它可使当前文本显示时向页面的中间对齐。

(7) <LEFT>标签。它可使当前行文本显示时向页面的左边界对齐。

(8) 注释标签。HDML 的注释标签和 HTML 的注释标签相同,可采用“<!-- comment_text -->”的形式,也可以去掉感叹号(!),直接用“<-- comment_text -->”的形式来注释。

下面就是利用文本类标签的一个简单例子:

```
Content-type: text/x-hdml; charset=GB2312
```

```
<HDML VERSION="4.0">
```

```
<!-- This is a simple card.-->
```

```
<DISPLAY name="card1">
```

```
<right>Welcome to WAP World!<br>
```

```
WAP is the way touch you and me. <line>
```

```
WAP Browser.
```

```
</DISPLAY>
```

```
</HDML>
```

表10.2 常用保留字符和特殊字符的转义序列

字符	转义序列
<	<
>	>
"	"
&	&
\$	&dol;
回车	
任何 ASCII 字符	&#nn; (其中 nn 是 ASCII 代码)

HDML 与 WML/WMLScript 语言一样,也有给一些保留字符和特殊字符规定的转义序列的显示方式。例如,小于号(<)的转义序列为“<”,由于小于号(<)不能直接写在要显示的文本中,所以要使用“<”来代替它。这样,当我们要显示“3<5”时,就应当写成“3<5”的文本形式才行。表 10.2 给出了常用保留字符和特殊字符的转义序列。

下面的程序给出了使用转义序列的例子,大家可以运行一下,看看实际效果。

```
Content-type: text/x-hdml; charset=GB2312
```

```
<HDML VERSION="4.0">
```

```
<DISPLAY name="card1">
```



```

<!--下面的一行显示 “3+5 >6 ” -->
<center> 3+5 &gt; 6<br>
<!--下面的一行显示 “ 3” is a odd number.” -->
<center>  &quot;3&quot; is a odd number.<br>
<!--下面的一行显示 “mouse & cat: $30.” -->
<center> mouse &amp; cat: &dol;30.<br>
</DISPLAY>
</HDML>

```

10.4.2 超链接标签

超链接标签可以实现 HDML 页面中多个卡片之间的跳转，我们通过下述例子来分析超链接标签的用法：

```

<HDML VERSION="4.0">
  <DISPLAY name="card1">
    <ACTION TYPE=ACCEPT TASK=GO DEST=#card2>
      This is CARD1
    </DISPLAY>

  <DISPLAY name="card2">
    <ACTION TYPE=ACCEPT TASK=GOSUB DEST=http:// tempwap.mywap.com/abc.html>
      This is CARD2
    </DISPLAY>
</HDML>

```

在这个程序中，我们建立了两个卡片，其中包含有<ACTION>标签。HDML 的<ACTION>标签其实相当于 HTML 中实现超链接功能的<A>标签，第 1 个<ACTION>标签语句<ACTION TYPE=ACCEPT TASK=GO DEST=#card2>执行的任务相当于 HTML 中的，即跳转到当前卡片组(即当前 HDML 页面)中名为 card2 的卡片，这是由 DEST=#card2 指定的，而跳转操作则是由 TASK=GO 确定的，跳转发生的条件由 TYPE=ACCEPT 指定，即当用户在浏览器选择了 ACCEPT 键、“确定(OK)”或与此类似的选项后发生。

在 HTML 中是执行跳转到当前页面的 card2 标记处。显然，<ACTION>

标签的 DEST 属性相当于 HTML 中<A>标签的 HREF 属性。

程序中的第 2 个<ACTION>标签语句<ACTION TYPE=ACCEPT TASK=GOSUB DEST=http://tempwap.mywap.com/abc.html> 相当于 HTML 中的，即指定当屏幕跳转到 card2 以后再按 ACCEPT 键(或类似的键及选项)则跳转到另一个卡片组 http://tempwap.mywap.com/abc.html，这相当于 HTML 中的另一个页面。在 HTML 中，浏览器根据用户的鼠标操作发生页面跳转，而 HDML 里则是根据 TYPE 属性所指定的值来触发向卡片或卡片组的跳转。<ACTION>标签 TYPE 属性可以接受的值实际上是用户所用的键盘(如手机按钮)或 LABEL 值，LABEL 值指手机等无线设备的软键(Softkey)在屏幕上显示的名称。TYPE 属性的取值如表 10.3 所示。

表10.3 TYPE属性的常取值

TYPE 取值	类别	可实现的任务
ACCEPT	LABEL 值	显示下一页或指定页
HELP	LABEL 值	如果手机内置卡里有当前页的帮助信息则显示
PREV	LABEL 值	显示上一页，如没有上页则取消当前 activity
SOFT1	LABEL 值	另行指定或无效
SOFT2	LABEL 值	另行指定或无效
SEND	LABEL 值	发送用户提交的信息给服务器
DELETE	LABEL 值	如果在可输入文本的卡片中，该值起删除作用

<ACTION>标签中的 TASK 属性用于指定将要执行的操作任务，具体用法后面再讲。我们这里可以把它看成与 HTML 中<FORM>标签的 ACTION 属性相类似的属性。

需要强调的是，卡片(由<DISPLAY>标签包含)中和卡片组(由<HDML>标签包含)中都可以定义使用<ACTION>标签。如果它们同时定义并且触发键相同，则当前卡片中的<ACTION>标签将覆盖掉卡片组的<ACTION>标签，即浏览器按照当前卡片中<ACTION>标签定义的操作执行。

10.4.3 图像显示标签

HDML 用于图像显示的标签也是，它和 HTML 中标签的用法完全一样(参见 10.2.2 节)，只是 HDML 显示的图像必须是 1 位的 BMP 图像，而且还是要手机支持的图像。

例如，我们想在手机屏幕上显示一个名为 abc.bmp 的并为手机支持的图像，假设该图



像位于 <http://wap.xyz.com>, 则 HDML 代码如下:

```
<HDML VERSION="4.0">
  <DISPLAY>
    <!-- 这里显示一幅 1 位的 BMP 图像 -->
    <IMG SRC="http://wap.xyz.com/abc.bmp" ALT="bmp" ><br>

    <-- 这里显示一幅浏览器内置的图标 -->
    <IMG ICON="smileyface" ALT=":-)" >
  </DISPLAY>
</HDML>
```

在 HDML 中, 标签的 ALT 属性用于指定当想要显示的图像无法显示时, 浏览器显示的信息; 而 SRC 属性则和 HTML 的 SRC 属性一样, 用于指定图像文件所在的路径; 标签还有 1 个 ICON 属性, 它用于指定手机 ROM 里预先存储的图像的名字, 如果 ROM 里没有则到 WAP 服务器中去寻找, 比如 UP.Link Sever 提供了 175 个可用图标, 具体的图标名字可参考各种 WAP 服务器的开发文档。

10.4.4 选单标签

HDML 中的选单功能类似于 HTML 中的 select 单选表单, 显示时是一组竖排的单选按钮表单。用户可以使用手机上的上、下键进行选择, 然后按 ACCEPT 键或其他指定的键予以确定。确定后, 浏览器即会按照选单标签的 TASK 属性所指定的任务去执行。

我们通过下述简单的例子来分析选单标签的用法和属性取值:

```
<HDML VERSION="4.0">
  <ACTION TYPE=ACCEPT TASK=GO DEST=#other_card>
  <CHOICE>
    <CE TASK=GO DEST=#card1>Your Choise1...
    <CE TASK=GO DEST=#card2>Your Choise2...
    <CE TASK=GO DEST=#card3>Your Choise3...
  </CHOICE>
</HDML>
```

该程序中的 <CHOICE> 标签和 <CE> 标签就相当于 HTML 中的 <SELECT> 标签和

<OPTION>标签，分别用于定义列表和列表中的选项。不同之处在于，<CE>标签里可以加入属性，而 HTML 中的<OPTION>标签一般没有直接的动作属性。

<CE>标签的 TASK 属性用于指定执行的操作任务，DEST 属性用于指定目的地，这里是选择当前项后浏览器要跳转的卡片。<CE>标签后面可以紧跟着写上想要显示的文本，如“Your Choise1...”等。

而且，HDML 规定<CE>标签的 TASK 属性所定义的动作优先执行，不论是卡片组还是卡片中<ACTION>标签的 TASK 属性指定了什么动作，<CE>标签 TASK 属性的动作均优先于它们的动作。

另外，利用选单我们也可以获得用户的选择值，注意下面的几行语句：

```
<CHOICE KEY=var1 IKEY=var2>
  <CE VALUE=1 LABLE=1>Choice 1
  <CE VALUE=2 LABLE=2>Choice 2
  <CE VALUE=3 LABLE=3>Choice 3
</CHOICE>
```

这里的<CHOICE>标签使用它的 KEY 属性来指定了接收<CE>标签的 VALUE 值的变量 var1，IKEY 属性所指定的变量 var2 用来接收被选选项的序号。这就是利用选单获取用户选择值的基本方法。KEY 属性和 IKEY 属性指定的变量可以在 HDML 程序的其他地方引用或处理。有关 HDML 变量的定义和引用方法我们下面马上讲到。

10.4.5 行为(ACTIVITY)及其嵌套

HDML 的网络应用都是围绕着行为(ACTIVITY)来组织的。ACTIVITY 与用户想要执行的 TASK 任务有关，每个 ACTIVITY 可以包含多个步骤(STEP)，这种处理模式是针对手机的特殊性而建立的，因为手机屏幕比较小，内存也比较小，所以一次不可能执行太多的任务，也不能显示太多的内容。而通过 ACTIVITY 和 STEP，我们可以把一个较大的任务分解为若干步骤来执行，每次 STEP 所执行的内容都能为手机所承受。例如，在邮件服务中有编辑新邮件、查看收件箱等 ACTIVITY，就以编辑新邮件这个 ACTIVITY 来说，它可能要通过写入收件人地址、邮件主题和邮件内容等一些步骤来完成，那么我们就可以把这些步骤通过一个一个的 STEP 来完成，每个 STEP 需要显示的内容均不超出手机浏览器屏幕。



在 HDML 页面中, 一般一个 STEP 就对应着一个卡片, 即一屏信息。例如, 在查询股票价格的 ACTIVITY 中, 第一个卡片即第一个 STEP 可以完成股票代码的输入, 第二个卡片也就是第二个 STEP 可以来完成股票价格的显示; 两个卡片之间的连接则可以使用前面介绍的<ACTION>标签中的属性 TASK=GO 来实现。

ACTIVITY 可以多层嵌套, 例如在查看收件箱的过程中可以进行删除邮件的操作, 此时用 TASK=GOSUB 可以转到删除邮件 ACTIVITY 的第一个 STEP 的卡片。在嵌套 ACTIVITY 中, 手机会保存 ACTIVITY 的运行状态, 包括各变量取值和卡片的历史记录, 而用户如果取消了当前的 ACTIVITY, 则浏览器会根据保存的状态从当前 ACTIVITY 返回到嵌套它的那个 ACTIVITY 的卡片。

嵌套的 ACTIVITY 间是可以传递参数的, 不过这要借助于变量来实现, 具体方法等后面讲解了 HDML 的变量以后再介绍。

10.4.6 变量的定义与引用

HDML 中变量的作用类似于 HTML 表元素中 VALUE 属性的作用。在 HDML 中, 用户可以自定义变量名, 变量名是严格区分大小写的, 而且一些特殊字符 “\$”、“<”、“>”、“=”、“/”、“\”、“&”、“*”、“#” 等不能用做变量名, 这些规定与 WML 和 WMLScript 语言有关变量命名的规定是一致的。为了节省手机的存储空间, 我们建议变量命名时应尽量取较短的变量名。HDML 规定, 变量的应用范围是定义该变量的 ACTIVITY 范围中。

HDML 中的变量是使用<ACTION>标签的 VARS 属性进行定义的, 该属性定义变量的形式为:

VARs=变量名 1=值&变量名 2=值&变量名 3=值&……

下面的例程中给出了定义变量的例子:

```
<HDML VERSION="4.0">
  <DISPLAY>
    <ACTION TYPE=ACCEPT TASK=GO DEST=#CARD2
      VARS=var1="Hello!"&var2=16.38>
    Press OK to view variable values.
  </DISPLAY>

  <DISPLAY NAME=CARD2>
```

```
<WRAP>$var1
<WRAP>$var2
</DISPLAY>
</HDMML>
```

该程序中第 1 个卡片中的<ACTION>标签中使用 VARS 属性，定义了两个变量：var1 和 var2，并给它们赋了初值。其中，var1 赋值为"Hello!"，同时说明它的类型为字符串型；var2 赋值为 16.38，同时说明它的类型为浮点型。

第 2 个卡片 CARD2 中则使用“\$变量名”的形式来引用变量，这一点与 WML 和 WMLScript 语言也是极为相似的。

10.4.7 获取用户输入

HDML 中用于获取用户输入数据的标签是<ENTRY>标签，它相当于 HTML 表单中的<INPUT>标签。<ENTRY>标签比较常用的属性有 4 个，它们的名称及作用如下：

(1) KEY 属性。该属性用于指定一个变量名，同时，<ENTRY>标签取得的用户输入就存储在该变量中。

(2) NAME 属性。它指定的是卡片的名字，作用是方便其他卡片引用这里的用户输入。

(3) DEFAULT 属性。用来定义用户输入数据的默认值。

(4) NOECHO 属性。该属性用于指定用户输入是否是密码形式，即当 NOECHO=TRUE 时，就用星号(*)来替代用户输入的字符，从而起到保密的目的。该属性的默认值是 FALSE，即不用星号(*)替代，直接显示用户输入的字符。

下面的程序给出了 HDML 中解决用户输入的方法：

```
<HDMML VERSION="4.0" MARKABLE=false>
  <ENTRY NAME=CardName KEY=usrName>
    <ACTION TYPE=ACCEPT TASK=GO DEST="#CARD2">
      Your name:
    </ENTRY>

  <DISPLAY NAME=CARD2>
    Hello $usrName
  </DISPLAY>
```



</HDML>

其中第 2 行<ENTRY>标签句<ENTRY NAME=CardName KEY=usrName>使用 NAME 属性定义了当前卡片的名字为 CardName，使用 KEY 属性定义了接受输入的变量名为 usrName。第 3 行<ACTION>标签则指定获取用户输入跳转到另一卡片，即 CARD2，该卡片将显示变量 usrName 的值，即用户刚输入的内容。

10.4.8 ACTIVITY 间的参数传递

在 ACTIVITY 间传递参数的基本原则如下：

(1) 如果 ACTIVITY 1 嵌套 ACTIVITY 2(更多的嵌套情况可以依此类推，下同)，那么 ACTIVITY 1 可以通过<ACTION>标签或<CE>标签来实现参数的传递，它们的形式为：

<ACTION TASK=GOSUB VARS=varname=value>

或 <CE TASK=GOSUB VARS=varname=value>

各标签均使用 VARS 属性定义了存储参数值的变量，这里是 varname，同时被赋值为 value，那么使用定义的该变量 varname 即可将参数值 value 传递给 ACTIVITY 2。

(2) 如果 ACTIVITY 1 嵌套 ACTIVITY 2，则 ACTIVITY 1 可以使用<ACTION>标签或<CE>标签的 RECEIVE 属性来指定变量，并接受从 ACTIVITY 2 返回过来的变量值。另一方面，ACTIVITY 2 可以用<ACTION>标签或<CE>标签的 RETVALS 属性来将这些变量值返回给 ACTIVITY 1。

RECEIVE 属性的语法形式为：

<ACTION RECEIVE=var1;var2>

或 <CE RECEIVE=var1;var2>

如果不需要传递 var1，则可用<ACTION RECEIVE=;var2>或<CE RECEIVE=;var2>的形式。

RETVALS 属性的语法形式为：

<ACTION TASK=RETURN RECEIVE=var1;var2>

或 <CE TASK=RETURN RECEIVE=var1;var2>

(3) 如果 ACTIVITY 1 嵌套 ACTIVITY 2，那么 ACTIVITY 2 可以用<ACTION>标签或<CE>标签的 CLEAR 属性来释放 ACTIVITY 1 的变量，CLEAR 属性的值应取 TRUE。语法

形式为:

<ACTION TASK=RETURN CLEAR=TRUE>

或 <CE TASK=RETURN CLEAR=TRUE>

上述形式是需要返回值(TASK 属性值为 RETURN)的情况下的语法, 如果不需要返回, 而是直接取消操作, 则可以采用 TASK 属性值为 CANCEL 的下述语法形式:

<ACTION TASK=CANCEL CLEAR=TRUE>

或 <CE TASK=CANCEL CLEAR=TRUE>

10.4.9 TASK 属性的取值

学习了 ACTIVITY 和<ACTION>、<CE>标签之后, 我们就可以总结一下 TASK 属性的可取值了。TASK 属性一般多用在<ACTION>标签和<CE>标签中, 我们介绍的它的取值也主要是针对这两种标签的。TASK 属性的取值如表 10.4 所示。

表10.4 TASK属性的取值

TASK 属性的取值	实现的任务
GO	转到 DEST 属性指定的 URL 地址或者卡片
GOSUB	转到 DEST 属性指定的嵌套 ACTIVITY
RETURN	从嵌套 ACTIVITY 返回调用它的卡片
CANCEL	取消当前的 ACTIVITY
PREV	返回前一个卡片
CALL	拨打 NUMBER 属性指定的电话号码
NOOP	空操作(什么也不做)

10.4.10 HDML 的 CGI 编程

总的来说, HDML 编程是符合 HTTP 协议和 CGI 标准的。我们可以在<ACTION>标签和<CE>标签中通过 METHOD 属性指定传递方式为 GET 或者 POST, 使用 POSTDATA 属性指定需要发送(POST)的数据。CGI 程序在接收时可按照 HDML 的标准方式获得数据, 输出时可将“Content-type:”指定为“text/x-hdml”。

HDML 的 CGI 编程与用户所用的开发环境有很大关系, 不同的开发环境采取的处理方法不尽相同, 所以我们这里无法一一述及, 请大家开发时根据所用的开发环境, 如 Phone.com 公司提供的 UP.SDK 开发环境, 具体参考它们提供的技术资料 and 程序库。



本章小结

本章我们主要学习了 HDML 编程，由于 HDML 语言与 HTML 语言用法类似，所以我们先用较大的篇幅介绍了 HTML 语言，然后通过比较 HTML 与 HDML 的标签、属性等方面的特点，来讲解 HDML 的编程知识。

对于 HDML 语言，大家不仅要了解它的开发环境的建立方法、页面文件的组成结构，还要重点掌握它的各种常用的标签，如文本标签、超链接标签、图像显示标签、选单标签等。HDML 提出了行为(ACTIVITY)的编程方法，大家应当了解 ACTIVITY 的处理思想和具体方法，尤其要掌握 ACTIVITY 的嵌套规则和 ACTIVITY 间参数的传递方法。大家还要掌握 HDML 的变量的定义与引用方法、获取用户输入和选单选择的方法等。

我们特别指出，HTML 语言是一种极为基本的网页设计和编程语言，WML 和 HDML 语言与它在很多方面都有相似之处，因此，掌握 HTML 语言不仅能够设计普通的网页，而且对于我们融会贯通地掌握 WML 和 HDML 语言，开发 WAP 应用等都会大有裨益。这也是我们拿出专门的篇幅介绍 HTML 语言的原因，希望能够引起大家的注意。



第 11 章 WAP 编程与开发的高级技术

第 11 章 WAP 编程与开发的高级技术

本章讨论 WAP 编程与开发的高级技术,主要包括 WAP 页面中汉字与图像的使用问题、HTML 过滤器和 HTML 页面向 WAP 页面的自动转换问题,以及结合使用 WAP 编程语言与 ASP、PHP、Perl、C、JSP、Servlet 等技术开发动态 WAP 页面的高级方法。本章涉及的技术知识比较多,读者需要具备 ASP、PHP、Perl、C、JSP、Servlet 以及 Web 数据库等方面的基础。这些方面的内容比较多,本书不可能详细讲解,我们只是围绕 WAP 的 WML 编程,讲解应用这些开发工具的方法。更深入的应用和开发工作还有待读者去探索和实现。

11.1 汉字与图像的使用问题

前已论述,在 WAP 网页中可以使用汉字和图像,但由于 WAP 的客户端设备具有内存少、屏幕小的特点,所以应用汉字和图像时必须进行适当的处理。下面我们就介绍具体的处理方法。

11.1.1 汉字使用与字符集转换

目前,由于 WAP 技术及其应用还没有完全成熟,不同的系统,不同的 WAP 服务器和 WAP 客户端设备执行的标准在细节上可能有所出入,所以 WAP 文件的编码方式、微型浏览器所支持的编码方式、服务器和微型浏览器的相关设置以及开发系统所支持的编码方式等都有可能存在一些差异。因此,当在 WAP 开发中,比如 WML 编写的页面中使用汉字的时候,一方面我们需要设置服务器和浏览器的配置,使之支持汉字字符集,另一方面我们需要对汉字字符集进行转换,转换成绝大多数 WAP 服务器、微型浏览器所支持的字符集(如 UTF8 或 UNICODE)或编码类型。



现在,使用最普遍的汉字字符集是 GB2312,那么我们为了顺利地在 WML 编程和 WAP 页面浏览中使用汉字,就需要进行以下工作:

(1) 在为 WAP 服务器中设置文件类型的 MIME 表时,可在“text/vnd.wap.wml”后加上“;charset=charset_name(字符集名称)”的内容。比如,若使用 GB2312 汉字字符集,则可输入 wml 文件的内容类型为“text/vnd.wap.wml; charset=gb2312”,这样就可使 WAP 服务器支持符合字符集 GB2312 的汉字。

(2) 在设计 WAP 网页时,可在程序中为返回类型指明汉字使用的字符集,即将 charset 加在向用户浏览器发送信息的类型后,比如使用 GB2312 的字符集,则可按以下方式写出程序代码:“Content-type: text/vnd.wap.wml; charset=gb2312”(不包括引号)。

如果使用 ASP、PHP 等技术编写 WML 程序,则可在相应的 ContentType 脚本语句的最后加上“;charset=gb2312”,指定汉字字符集。

(3) 使用专门的字符转换工具来转换,将编写的 WML、WMLScript 或 HDML 的程序代码转换为 UTF8 或 UNICODE 的编码。目前 Internet 有许多站点提供有 WAP 编程中所需的字符转换工具或控件,不仅可以转换汉字,而且还可以转换繁体中文、韩文、日文等其他字符集。

例如, www.WapSchool.com 和 www.BookingAll.com 提供有一个可免费下载的小巧玲珑、简单而又功能强大的字符集转换工具 converter.class,它可以将使用汉字的 WAP 程序转化为任何类型的字符集。读者从本书所随光盘中也可以找到该转换工具。它使用 Java 语言编写,采用的是命令行执行方式,安装时需要将 converter.class 文件复制到所用系统的 CLASSPATH 目录中。该工具软件可在任何平台运行。它的用法如下:

```
java converter filename convertingType
```

其中, filename 是需要转换的文件名, convertingType 是要转换的目的字符集类型。

例如,我们需要将 WML 程序文件 mytest.wml 转换成 unicode 或 UTF8 编码字符集,则可分别使用如下命令:

```
java converter mytest.wml Unicode
java converter mytest.wml UTF8
```

当然,用户所用机器系统中需要有 Java Runtime Environment 环境,本书所随光盘提供了 Java Runtime Environment 1.1,读者可以安装该软件进行测试。

11.1.2 图像使用与图像格式转换

WAP 网页只支持 1 位的 bmp 位图,即单色的 bmp 图像,普通图像不能直接应用到 WAP 页面中,需要使用一些专门的工具软件进行格式转换,将图像转换为 WAP 支持的 wbmp 格式,然后才能应用到 WAP 页面中。目前 Internet 上也有许多站点提供了这类转换工具的免费下载服务,例如,大家可以到 <http://wap.gingco.de> 或 <http://www.gingco-newmedia.de> 去下载一个名为 pic2wbmp 的软件控件,它能够将其他类型的图像转换为 WAP 支持的 wbmp 格式。本书随书光盘中提供了该软件,可供大家使用。

转换图像格式的操作方法很简单。首先启动 pic2wbmp,命令方式为:

```
java -classpath .;pic2wbmp.zip;JimiProClasses.zip;%CLASSPATH% pic2wbmp
```

或者使用 pic2wbmp 软件包提供的批处理文件 pic2wbmp.bat,也可启动 pic2wbmp。

接下来,从 pic2wbmp 运行后的窗口中单击“Browse”按钮,并从出现的对话框中选择想要转换的图像文件,然后返回到 pic2wbmp 窗口,从菜单中选择“Save WBMP as”命令,即可将当前选中的图像文件另存为 WBMP 格式的文件。这种文件的扩展名为“.wbmp”。

pic2wbmp 也需要 Java Runtime Environment 1.1 或更新版本的支持,用户所用系统中必须安装 Java Runtime Environment。本书所附光盘中提供了这一软件,供大家使用。

将所需的图像文件转换为 wbmp 格式后,就可以把它应用到 WAP 网页中来了,具体用法我们在讲 WML、WMLScript 及 HDML 编程时已经介绍过,这里不再重述,大家可以参考前面的相关内容。

11.2 ASP 和数据库技术在 WAP 开发中的应用

利用 ASP 技术可以开发出动态 WAP 页面,将 ASP 技术和数据库技术相结合,则可以开发更为复杂的 WAP 应用。有关 ASP 和 Web 数据库的知识请读者参考专门书籍,我们这里以 WML 为例,说明 ASP 和数据库技术在 WAP 开发中的应用。

11.2.1 在 WML 程序中使用 ASP

在 WML 程序中使用 ASP 的规则很简单,只要做到以下 5 条即可:



(1) 声明 ASP 采用的脚本语言。与 ASP 中声明的方式类似，格式为：

```
<% @ Language=VBScript %>
```

(2) 声明 wml 文件类型。这可采用 ASP 的 Response 对象，并将 ContentType 作为它的方法，书写时使用<%和%>包含起来，格式为：

```
<% Response.ContentType="text/vnd.wap.wml" %>
```

(3) 在 WAP 服务器端，增加服务器对 ASP 的处理能力。这可在 IIS 中选中 WAP 站点或其主目录，并单击鼠标右键，从出现的菜单中选择“属性”命令，打开它的对话框，选择其中的“虚拟目录”选项卡。这里面有一项应用程序设置，将“许可”一项设置为“执行(包括脚本)”即可。更详细的方法可参考我们第 4 章的内容。

(4) 其余语句和形式采用 WML 的编程方法即可。

(5) 程序文件保存时要采用“.asp”的扩展名。

例如，下面就是一个在 WML 程序中应用 ASP 的简单例子：

```
<% @ Language=VBScript %>
<% Response.ContentType="text/vnd.wap.wml" %>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="main">
    <p>
      Hello!
    </p>
  </card>
</wml>
```

这是一个显示“Hello!”的程序，可把它保存为 hello.asp。注意，文件扩展名一定是.asp 才行，它与 hello.wml 是两个不同的文件。我们可以把该文件放到或上传到 WAP 服务器的相应目录中，比如 wap 目录。然后，我们启动模拟浏览器，比如 WinWap，并在 URL 栏里输入“http://127.0.0.1/wap/hello.asp”并按回车，即浏览使用 ASP 的 WML 页面的效果。

11.2.2 在 ASP 中编写 WML 程序

我们也可以直接在 ASP 程序中编写 WML 程序，并可把它输出为 WML 程序。具体的处理规则如下：

- (1) 将 WML 程序使用<%和%>包含起来，格式为：

```
<%  
.....(语句)  
%>
```

- (2) 声明 wml 文件类型。与前面讲述的方法相同，可采用 ASP 的 Response 对象，并将 ContentType 作为它的方法，但书写时不再需要使用<%和%>。格式为：

```
Response.ContentType="text/vnd.wap.wml"
```

- (3) 对于其他的 WML 语句，一律采用 Response 对象的 write 方法来处理。基本格式为：

```
Response.write(".....(WML 标签或语句)")
```

其中的“WML 标签或语句”在 write 方法中可以连写，以节省程序代码行。

- (4) 在 WAP 服务器端，增加服务器对 ASP 的处理能力。操作方法与前面介绍的一样，这里不再重述。

- (5) 程序文件保存时也要采用“.asp”的扩展名。

下面的程序就是在 ASP 中编写 WML 页面的简单例子：

```
<%  
    'msg = "Hello "  
    Response.ContentType = "text/vnd.wap.wml"  
    Response.write("<?xml version='\"1.0\"'?><!DOCTYPE wml PUBLIC \"\"-//WAPFORUM//DTD  
WML 1.1//EN\"\" \"http://www.wapforum.org/DTD/wml_1.1.xml\"\">")  
    Response.write("<wml><card>")  
    Response.write("hello wml ! test ok !")  
    Response.write("</p></card></wml>")  
%>
```

当然，我们这里给出的属于 ASP 与 WML 结合使用的简单例子，但采用这种基本方法，



并利用 ASP 的复杂功能，便可以开发出能够实现丰富功能的 WML 页面。

例如，下面的 ASP 程序不仅可以显示“Hello”信息，同时可以显示信息的来源地或所在环境，这是通过利用 Request 对象的 ServerVariables 方法来获取服务器的 HTTP_X_UP_SUBNO 和 HTTP_X_UP_UPLINK 两个变量实现的。程序代码如下：

```
<%  
' hellowml.asp  
,  
  
Dim msg, subId, uplink  
  
msg = "Hello "  
subId = Request.ServerVariables("HTTP_X_UP_SUBNO")  
uplink = Request.ServerVariables("HTTP_X_UP_UPLINK")  
  
If Not Len(subId) = 0 Then  
    msg = msg & "from " & subId  
End If  
If Not Len(uplink) = 0 Then  
    msg = msg & " at " & uplink  
End If  
  
Response.ContentType = "text/vnd.wap.wml"  
Response.write("<?xml version=""1.0""?><!DOCTYPE wml PUBLIC ""-//WAPFORUM//DTD WML  
1.1//EN"" ""http://www.wapforum.org/DTD/wml_1.1.xml"">")  
Response.write("<wml><card><p>")  
Response.write(msg)  
Response.write("</p></card></wml>")  
%>
```

为增强大家对 ASP 编写 WML 程序的印象，我们再给出 Phone.com 提供的一个例子。该例的作用是根据对服务器和客户端设置测试结果的不同，给出不同的警告信息。程序代码如下：

```
<%  
' pushAlert.asp  
,
```



```
Dim uplink, subId, url, ttl, alertType, title
Dim contentType, lastResult
contentType = "application/x-up-alert"

uplink = Request.ServerVariables("HTTP_X_UP_UPLINK")
subId = Request.ServerVariables("HTTP_X_UP_SUBNO")

url = "http://updev.phone.com/dev/hdml/devhome3.hdml"
ttl = 3600
alertType = "D---"
title = "Call: " & subId

Set NtfyCl = Server.CreateObject("Ntf3Client.Ntf3Client.1")
NtfyCl.NtfnSetHost uplink
NtfyCl.NtfnSetTimeout ttl
NtfyCl.NtfnPostAlert subId, url, ttl, alertType, title
lastResult = NtfyCl.NtfnGetLastResult

Dim errString, msg
If lastResult <> 0 Then
    errString = NtfyCl.NtfnGetErrorDetail
    If errString <> NULL Then
        msg = errString
    Else
        msg = "No error detail, error number: " & lastResult
    End If
Else
    msg = "Sent alert to " & subId & " at " & uplink
End If

Dim my, digestString
set my = Server.CreateObject("PDCCDigestUtils.CPDCCDigest")
my.DeckType = 0
my.DigestAddDeck "deck1", "<wml><card><p>" & msg & "</p></card></wml>"

my.DigestSuppressContentType 1
digestString = my.DigestSerialize(0)
Response.ContentType = my.DigestGetContentType
```



```
Response.BinaryWrite(digestString)
%>
```

为便于大家学习，我们本书所附光盘中都提供了这些程序的源代码，大家不要忘了实际测试一下。

11.2.3 利用 ASP 在 WML 中实现动态数据库应用

通过前面介绍的例子，我们可以看出利用 ASP 技术可以为 WML 实现非常复杂的应用。我们知道，ASP 的数据库处理能力是很强的，所以，利用 ASP 我们可以在 WML 中实现动态的数据库应用。由于举例说明这种应用需要占用很大的篇幅，所以我们这里只给出基本的开发步骤：

第一步，分析 WAP 的实际问题，设计出所需的数据库，并建立其中的数据表。例如，我们可以使用 Microsoft Access 等数据库系统来完成这一方面的工作。

第二步，建立 ODBC 源，并对上述数据库进行命名。操作方法十分简单：以 Windows 95/98/NT/2000 为例，我们只需在“控制面板”中双击“ODBC 数据源”图标，从打开的“ODBC 数据源管理器”对话框中即可完成相关操作。

第三步，根据实际问题，编写引用和处理数据库的 ASP/WML 程序。

第四步，将程序上载到 WAP 服务器，随后就可以进行测试。通过测试后，就可以为 WAP 用户使用了。

11.3 PHP 编程在 WAP 开发中的应用

我们可以使用 PHP 来编写 WAP 程序，比如开发动态的 WML 页面，不过条件是需要将输出的标签或语句限制在 WAP 微型浏览器可接受的范围之内。而且，PHP 还可以在一个 HTML(及 HDML)文件中编写出既适合于 HTML(及 HDML)，也适合于 WML 的内容。PHP 的源代码对于 WAP 客户端来说是不可见的，所以我们可以针对 HTML 浏览器输出 HTML 页面，而针对 WAP 浏览器输出 HDML 页面或 WML 页面。

11.3.1 基本规则

以 WML 为例，使用 PHP 编写 WML 程序的基本规则如下：

(1) 使用 “<?” 和 “?”>” 来包含 PHP 编写的 WML 程序行，格式为：

```
<?
.....(程序行);
?>
```

(2) 程序行语句均以分号(;)结尾，这是与 ASP 编写 WML 程序不同的地方。

(3) 使用 PHP 的 header 关键字来声明 WML 的文件类型，基本格式为：

```
header("Content-type: text/vnd.wap.wml");
```

上述格式是专为 WAP 浏览器识别而采取的书写格式。如果开发中用户想使用普通的浏览器，如 IE 浏览器来测试程序效果，则可在该语句的前面加上双斜线(//)，格式为：

```
// header("Content-type: text/vnd.wap.wml");
```

这样，基于 PC 的浏览器将忽略程序中无法理解的 WML 标签，这时当前的程序实际上成为了 HTML 页面。当想在 WAP 设备或者模拟器上测试的时候，只需要去掉“//”，当前页面就会自动变成 WML 页面。

(4) 其他 WML 标签和语句行一律使用 PHP 的 echo 关键字来声明，每个 WML 行的后面加上换行符 “\n” (根据需要，也可加多个)。其基本格式为：

```
echo(".....(WML 标签和语句行)\n");
```

例如，下面的语句就是 PHP 编写 WML 语句行的典型例子：

```
echo("<?xml version=\"1.0\"?>\n");
```

而且，echo 语句中的“WML 标签和语句行”可以连写，以节省程序代码行。例如，下面的一行语句就连写了 WML 的 3 行标签语句：

```
echo("<wml> <card> <p>");
```

(5) 声明脚本语言。这是对 PHP 编程的继承，向编译器声明当前程序采用的脚本语言为 PHP，格式为：



```
<script language="PHP">
```

不过这一句可以省略，因为编译器能够自动识别 PHP 的脚本程序。

(6) 文件最后保存时采用的扩展名为 “.php” 或 “.php3”，而不是 “.wml” 等扩展名。

根据上述规则，我们可以用 PHP 编写出 WML 页面的文件头，并让程序在 WAP 浏览器中显示 “Hello, World!”，代码如下：

```
<?
    header("Content-type: text/vnd.wap.wml");
    echo("<?xml version=\"1.0\"?>\n");
    echo("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\" \"http://www.
wapforum.org/DTD/wml_1.1.xml\">\n\n");
?>
<?
    echo("<wml> <card> <p>\n");
    echo("Hello, World!\n");
    echo("</p> </card> </wml>\n");
?>
```

11.3.2 程序举例

下面我们给出一个非常有用的 PHP 实现的 WML 应用，它称为 PizzaCalc，可以根据输入的 pizza 帐单及人数，算出每个人的花费。注意程序中根据 WML 的规则，对一些特殊符号，如双引号等使用了转义字符。该程序不仅可以进行变量处理，而且还可以传递相应的参数。程序代码如下：

```
// pizzacalc.html
<?
    header("Content-type: text/vnd.wap.wml");
    echo("<?xml version=\"1.0\"?>\n");
    echo("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"
'http://www.wapforum.org/DTD/wml_1.1.xml\">\n\n");
?>

<?
    echo("<wml>\n");
```

```
?>

<?
if($action == "calc") {
    echo("<card id=\"result\" title=\"PizzaCalc\">\n");
    echo("<do type=\"prev\" label=\"Back\">\n");
    echo("<go href=\"pizzacalc.html#input\"/>\n");
    echo("</do>\n");
    echo("<p>\n");
    echo("The cost per eater will be ".$total / $eaters."<br/>\n");
}
else {
    echo("<card id=\"input\" title=\"PizzaCalc\">\n");
    echo("<p>\n");
    echo("<anchor>Split Pizza bill <go href=\"pizzacalc.html?total=\".$(total)&eaters=\".$(eaters)&action=
calc\"/></anchor>\n");
    echo("<br/>\n");
    echo("Total cost: <input type=\"text\" name=\"total\" format=\"%N\"/>\n");
    echo("Eaters: <input type=\"text\" name=\"eaters\" format=\"%N\"/>\n");
}
?>

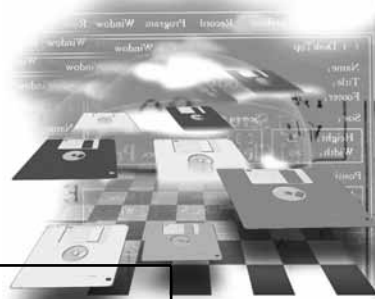
<?
echo ("</p> </card> </wml>\n");
?>
```

11.4 Perl 与 WAP 的综合应用

使用 Perl 也可生成 WAP 页面和开发出复杂的 WAP 应用。我们还是以 WML 为例来说明 Perl 在 WAP 开发中的具体应用。当然，用户首先应当配置好自己的系统，使之能够运行 Perl，而且还要具有 CGI 的基本知识，了解 Perl 的语法。

11.4.1 基本规则

使用 Perl 编写 WML 程序的基本规则如下：



(1) 程序行语句均以分号(;)结尾,这是与 PHP 编写 WML 程序一致的地方。

(2) 使用 print 语句来声明 WML 的文件头信息,并在 WML 行后面加上换行符“\n”。

基本格式为:

```
print ".....(WML 标签或语句行)\n";
```

声明 WML 文件头信息的 Perl 代码为:

```
print "Content-type: text/vnd.wap.wml\n\n";
print "<?xml version=\"1.0\"?>\n";
print "<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\" \"http://www.wapforum.org/
DTD/wml_1.1.xml\">\n";
```

文件头信息的声明也可以使用 Perl 的标量变量来实现,具体方法下面马上介绍。

(4) 其他 WML 标签和语句行的实现比较灵活,既可以使用上述格式的 print 语句来实现,也可以通过 Perl 的标量变量形式来实现。我们这里主要介绍后一种形式,即将一个 WML 卡片的内容赋给一个 Perl 的标量变量,然后通过变量的操作来实现 WML 页面,其基本格式为:

```
$变量名 = 标识名 (
.....(WML 标签或语句)
);
```

或:

```
$变量名 =
'
.....(WML 标签或语句)
';
```

例如,显示“Hello!”信息的 WML 页面利用标量变量的第一种形式可以表示为:

```
$testwml = mywml (
<wml>
  <card id="main">
    <p>
      Hello!
    </p>
```

```
</card>
</wml>
);
```

其中标量变量名 `testwml` 和标识名 `mywml` 由用户定义，不过要符合 Perl 的语法要求。
上述页面利用标量变量的第二种形式可以表示为：

```
$testwml =
'<wml>
  <card id="main">
    <p>
      Hello!
    </p>
  </card>
</wml>';
```

(5) 使用 Perl 语言编写 WML 程序时，Perl 的注释符“#”仍然可以使用，且需要告诉系统 Perl 解释器所在的目录，如“`/usr/bin/perl`”，这一行通常放在程序的第一行：

```
#!/usr/bin/perl
```

其中的感叹号(!)用于表明该程序的类型，不可省略。

(6) 文件最后保存时采用的扩展名为“`.pl`”，如果实现的是 CGI 编程，则取扩展名为“`.cgi`”，不能取“`.wml`”等扩展名。

在这些基本规则下，我们就可使用 Perl 语言的强大 CGI 处理功能，来编写复杂的 WML 应用了。

11.4.2 程序举例

假设我们需要开发一个 WML 应用，WAP 用户可以通过职员及其电话号码列表来查询职员信息。现在我们暂时有两个职员的数据，但将来还需要增加职员数量。因此，我们需要将职员数据保存在一个文本文件中，以便将来增加数据。该文本文件中的数据可通过随后编写的动态 WML 页面来显示。

首先，我们建立一个名为“`people_data.txt`”的文本文件，来存放职员信息，内容如下：



```
# Name|phone num|fax num|Title
Cao Jian|123-4567|098-7654|Director of OS Development
Zhang San|123-4568|098-7655|Senior VP of OS Research
```

其中，“|”符号是用来区别字段的，“#”是用来注释数据结构的。

为了说明使用 Perl 来编写的必要性，我们先给出使用 WML 来显示这两名职员信息的卡片组，代码如下：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>

  <card title="Telephone Book" id="index">
    <p>
      Welcome to NEW Inc. Telephone Book
      <a href="#view">Click to view list</a>
    </p>
  </card>

  <card title="List of Names" id="view">
    <p>
      <a href="#cao_info">Cao Jian - 123-4567</a>
      <a href="#zhang_info">Zhang San - 123-4568</a>
    </p>
  </card>

  <card title="Info for Cao Jian" id="cao_info">
    <p>
      Director of OS development
      Phone number: 123-4567
      Fax number: 098-7654
    </p>
  </card>

  <card title="Info for Zhang San" id="zhang_info">
```

```
<p>
    Senior VP of OS research
    Phone number: 123-4568
    Fax number: 098-7655
</p>
</card>
```

```
</wml>
```

通过这个程序我们可以看到，如果增加新的职员，那么我们就不得不重新修改程序文件。这显然不是一个好办法。因此，我们设计了 people_data.txt 文本文件，可以随时增加职员数据，然后使用 Perl 实现上述 WML 页面的任务。我们的目标是，程序设计好后无需改动，只需通过给文本文件添加职员数据，就可以通过 Perl 编写的 WML 页面实现对增添职员后数据的浏览。

使用 Perl 编程时，最先需要完成的是设置我们的 Perl 环境。我们可以使用 CGI 模块来进行 HTTP 数据显示和处理，代码如下：

```
#!/usr/bin/perl -w
use strict;
use CGI qw/:standard/;

# Our CGI object
my $q = new CGI;
```

接下来，我们使用 Perl 语言编写上述 WML 页面的主体部分，同时建立标量变量，来保存文本文件名，以备后面从中读取数据。另外创建动态 ID，这里用 "a" 表示，用于标识每一条职员数据，即一条记录。我们的目的是一次性地将所有的职员数据都生成卡片，然后使用一个初始化的缓冲区来显示职员的数据内容。这段代码如下：

```
# 定义获取职员信息的数据文件
my $data = "people_data.txt";

# 显示标题的卡片
my $title_card = qq (
```



```

<card title="Telephone Book" id="index">
    <p>
        Welcome to NEW Inc. Telephone Book<br/>
        <a href="#view">Click to view list</a>
    </p>
</card>
);

# 初始化第一条记录的 ID
my $id = "a";

# 开始创建浏览卡片
my $view_card = '<card title="All names in directory" id="view">
<p>
Click on a name for more info<br/>';

# 用于临时存放卡片信息的变量(相当于缓冲区)
my $all_info_card;

```

下面，我们要打开数据文件，当然打开前要检查权限，以保证我们真的能够打开，然后进行数据的读取。读取时，指定文本文件中以“#”开头的行仅作注释之用，不属于被读的内容。在读取数据的过程当中，我们需要做一定的检查，以保证每个部分的确有数据。在这里，我们把从文件中提取的数据传给两个子过程，它们将返回由这些参数数据所确定的卡片。注意我们使用了 Perl 的“.= operator”运算符，它使得子过程返回的字符串将不断地扩充。实现这段任务的代码如下：

```

# 读取指定文件中的数据
open(FILE,"$data") || die "Can't open $data: $!\n";

while (<FILE>) {
    chomp;
    next if (/^\#/);
    my ($name, $phone, $fax, $position) = split(/\|/);

    # 调用子过程，创建浏览卡片
    $view_card .= build_view_card($name, $phone, $fax, $position);
}

```

```
# 调用子过程，创建职员信息卡片  
$all_info_card .= build_info_card($name, $phone, $fax, $position);  
}
```

其中调用的子过程 build_view_card 的代码如下：

```
sub build_view_card {  
    $id++;  
    my ($name, $phone, $fax, $position) = @_;  
    my $info_item = qq (  
        <a href="#$id">$name - $phone</a><br/>  
    );  
    return $info_item;  
}
```

它的功能是接收参数数据，并把这些数据插入到一个简单的模板里面。由于我们每次都在增加 \$id 变量的数值，从而保证每个记录都有唯一的 id 号。这是因为 Perl 允许我们直接使用 ++ 操作符，使得 \$id 变量的值从 'a' 增加到 'b'、'c' 等进行递增。

另一子过程 build_info_card 的代码如下：

```
sub build_info_card {  
    # 定义用于显示单个职员信息的卡片  
    my ($name, $phone, $fax, $position) = @_;  
    my $info_card = qq (  
        <card title="Info for $name" id="$id">  
        <p>  
        $position<br/>  
        Phone number: $phone <br/>  
        Fax number: $fax <br/>  
        <do type="prev"><prev/></do>  
        </p>  
        </card>  
    );  
    return $info_card;  
}
```

以上的分析和实现的代码完成了本例的基本功能，下面我们还要写出程序的 WML 文件头和执行上述代码的主过程，代码如下：

把以上使用 Perl 编写的 WML 代码放在一起(存放数据的文本文件内容除外),就是我们解决的动态显示职员信息的 WML 应用程序。

使用 C/C++ 也可生成 WAP 页面和开发出复杂的 WAP 应用。我们仍以 WML 为例来说明 C/C++ 在 WAP 开发中的具体应用。当然，用户首先应当在自己的系统中安装好诸如 Visual C++ 的开发系统，并使之能够正常运行，而且要具有一定的 CGI 基本知识，了解 C/C++ 的语法及函数。

11.5.1 基本规则

使用 C/C++ 编写 WML 程序的基本规则如下：

- (1) 程序行语句均以分号(;)结尾，这与 C/C++ 语言的原本要求是一致的。
- (2) 使用 printf 语句来编写(输出)WML 的标签或语句，并在 WML 行后面加上换行符“\n”。基本格式为：

```
printf(".....(WML 标签或语句行)\n");
```

如果 WML 标签或语句中有引号等特殊字符，则应采用 C/C++ 的书写规则，比如将引号(")加上斜杠“\”。

下面的语句就说明了使用 C/C++ 的 printf 编写 WML 程序的例子：

```
printf("Content-type: text/vnd.wap.wml\n\n");
printf("<?xml version='1.0'?'>\n");
printf("<!DOCTYPE wml PUBLIC '-//WAPFORUM//DTD WML 1.1//EN'");
printf("\ \"http://www.wapforum.org/DTD/wml_1.1.xml\">\n");
printf("<wml>\n");
printf(" <card id='datecard' title='MyWMLCC'>\n");
.....
```

与 C/C++ 的其他语句一样，WML 的语句也要放在 C/C++ 的函数中予以实现，后面的举例会说明这一点。

(3) 使用 C/C++ 语言编写 WML 程序时，C/C++ 的注释符“//”及“/* */”仍然可以使用，其他编程规则及预编译文件等均可采用。

(4) 文件最后保存时采用的扩展名为 C/C++ 程序文件及预编译文件的扩展名，不能再使用“.wml”等扩展名。

11.5.2 程序举例

熟悉 C/C++ 的读者都知道，利用 C/C++ 可以实现极为广泛和复杂的应用，但程序规模通常也比较大，因此我们这里的篇幅恐怕不允许我们讨论过于复杂的例子。下面我们只通



过一个比较简单的例子，说明使用 C/C++ 开发 WML 页面应用的方法。

我们这个例程利用 CGI 功能实现了 WAP 浏览器的动态输出效果。程序很简单，我们就不过多解释了。程序清单如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
void GenerateCGIHeader();
void GenerateCGIFooter();
/*-----*/
int main(int argc, char* argv[])
{
    char tmpBuf[128];
    char strGET[]="GET";
    if (getenv("REQUEST_METHOD") == NULL)
    {
        printf("This script is not intended to run from shell.\n");
        return -1;
    }
    // 检查 request 的方法(method)是否是 GET
    strcpy(tmpBuf, getenv("REQUEST_METHOD"));
    if (strcmp(tmpBuf, strGET) != 0)
    {
        printf("This script requires use of GET-method.\n");
        return -1;
    }
    /* 显示 WML 页面内容 */
    GenerateCGIHeader();
    /* 显示操作系统日期和时间 */
    _strtime( tmpBuf );
    printf( "OS time:%s\n", tmpBuf );
    _strdate( tmpBuf );
    printf( "OS date:%s\n", tmpBuf );
    /* 结束 WML 页面的显示 */
    GenerateCGIFooter();
    return 0;
}
```

```
}

/*-----*/
void GenerateCGIHeader()
{
    /* Generate header for CGI response */
    printf("Status: 200\n");
    printf("Content-type: text/vnd.wap.wml\n\n");
    printf("<?xml version=\"1.0\"?>\n");
    printf("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"");
    printf("\" \"http://www.wapforum.org/DTD/wml_1.1.xml\">\n");
    printf("<wml>\n");
    printf(" <template>\n");
    printf(" <do type=\"prev\" label=\"Back\">\n");
    printf(" <prev>\n");
    printf(" </do>\n");
    printf(" </template>\n");
    printf(" <card id=\"datecard\" title=\"CGIDate\">\n");
    printf(" <p>\n");

}

/*-----*/
void GenerateCGIFooter()
{
    /* Generate footer for CGI response */
    printf(" </p>\n");
    printf(" </card>\n");
    printf("</wml>\n");
}

/*-----*/
```

11.6 JSP 技术在 WAP 开发中的应用

使用 JSP(Java Server API)技术可以简单而快速地在 WAP 终端上生成动态的 WAP 页面，而且能够利用 JSP 的诸多功能，实现复杂的 WAP 应用。我们仍以 WML 为例，讲解 JSP 技



术在 WAP 开发中的应用。

11.6.1 基本规则

使用 JSP 技术前,首先要建立它的开发环境。一般是在 Windows NT 上安装并建立 Java Server Web Development Kit(JSWDK),其版本可为 1.01 版或更高的版本。

为了测试和浏览 JSP 实现的 WAP 应用,还需要安装 WAP 模拟器(浏览器),如 Nokia WAP Toolkit,并安装 Java Runtime Environment (JRE) 1.2.2 或以上版本。

完成以上准备工作之后,就可以按照以下的基本规则使用 JSP 编写 WML 程序了。

(1) WML 的标签和语句均可直接写在 JSP 程序中。

(2) 如果想使用 JSP 来处理 WML 的标签和语句,则需要使用“<%”和“%>”引起来,格式为:

```
<%  
.....(JSP 编写的 WML 语句行);  
%>
```

例如,以下几行语句就是利用 JSP 的 out.println 来输出 WML 的标签和语句:

```
<%  
    out.println("<p>");  
        out.println("Hello from script code!<br/>");  
    out.println("</p>");  
%>
```

(3) JSP 编写的 WML 语句要用引号(" ")引起来,然后再用括号括起来,而且后面还要加上分号(;),格式为:

JSP 对象.方法/关键字("WML 标签或语句行 ");

例如,下面就是反映 JSP 书写 WML 语句格式的例子:

```
out.println("Hello from script code!<br/>");
```

(4) WML 的文件类型可以使用 JSP 的 response 对象进行声明,也可以在声明页面语言时一块儿声明,它们的格式分别如下:

```
<% response.setContentType("text/vnd.wap.wml"); %>
```

或

```
<% @ page language="java" contentType="text/vnd.wap.wml" %>
```

(5) 最后的程序文件保存时要采用 “.jsp” 的扩展名，即保存为 JSP 的程序文件。JSP 将被编译成 Java 源文件，最后成为 servlet。

了解了以上基本规则，我们下面给出一个利用 JSP 向 WAP 浏览器显示“Hello from script code!” 信息的简单程序，通过该程序大家可以进一步认识利用 JSP 开发 WML 页面的具体方法。

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/
wml_1.1.xml">
<% response.setContentType("text/vnd.wap.wml"); %>
<wml>
  <card id="start">
    <do type="accept">
      <go href="index.jsp#test"/>
    </do>
    <p>JSP Test:<br/>
      Press accept to continue!<br/>
    </p>
  </card>

  <card id="test">
    <do type="prev">
      <prev/>
    </do>
    <%
      out.println("<p>");
      out.println("Hello from script code!<br/>");
      out.println("</p>");
    %>
  </card>
</wml>
```



11.6.2 程序举例

我们下面利用 JSP 和 WML 实现一个为移动用户定时更新约会的例子。该例的应用程序共包括两个页面。第一个页面的文件是 pick_appointment.jsp，它提供了一个选择卡片，当用户选择了其中某一个约会时间时，浏览器就会带着本次约会的 ID 号进入到第二个页面，即文件名为 show_appointment_data.jsp 的页面。我们在第二个页面编写了两个卡片，其中第一个卡片用于显示会面的时间，第二个卡片用于显示数据输入，让用户通过输入 ID 而取消约会。

程序中动态的约会数据是通过 Java Bean 的实例来取得的，具体过程其实是通过 JDBC 连接到数据库的过程。取消约会的操作是通过 servlet 实现的。由于用户可能随时取消某个约会，所以我们需要对 pick_appointment.jsp 页面进行定时刷新。下面我们就给出这一应用程序的源程序。

pick_appointment.jsp 页面文件的程序代码如下：

```
<%@ page language="java" contentType="text/vnd.wap.wml" %>

<jsp:useBean id="appointmentBean" class="mwebber.samples.AppointmentBean" scope="application" />

<%!
// 下面创建针对每次约会的选项<option>元素
private String getOptions(mwebber.samples.AppointmentBean appointmentBean) {
    StringBuffer sb = new StringBuffer();
    int[] appointmentIDs = appointmentBean.getAppointmentIDs();
    for(int i=0; i<appointmentIDs.length; i++) {
        sb.append("<option onpick=\"show_appointment_data.jsp?id=\"");
        sb.append(i);
        sb.append("\>");
        sb.append(appointmentBean.getAppointmentTime(i));
        sb.append("</option>");
    }
    return sb.toString();
}
%>
```



```
<?! String strXMLPrologue = "<?xml version=\"1.0\"?>"; %>

<%-- WML 内容开始 --%>

<%= strXMLPrologue %>

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

<wml>
    <card id="pick" title="Appointments">
        <!-- 每分钟刷新一次卡片组 -->
        <onevent type="ontimer">
            <go href="pick_appointment.jsp"/>
        </onevent>

        <timer value="600"/>

        <!-- 回显卡片 -->
        <do type="prev">
            <prev/>
        </do>

        <!-- 为要选择约会而显示"select" -->
        <p>
            <select title="Appointments">
                <%= getOptions(appointmentBean) %>
            </select>
        </p>
    </card>
</wml>

<%-- WML 内容结束 --%>

show_appointment_data.jsp 页面文件的程序代码如下：

<% @ page language="java" contentType="text/vnd.wap.wml" %>

<jsp:useBean id="appointmentBean" class="mwebber.samples.AppointmentBean" scope="application" />

<%
// 使用 request 对象的方法获取"id"参数的值
```



```
int intAppointmentID = Integer.parseInt(request.getParameter("id"));
%>

<%! String strXMLPrologue = "<?xml version=\"1.0\"?>"; %>

<%-- WML 内容开始 --%>
<%= strXMLPrologue %>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/
wml_1.1.xml">

<wml>
  <card id="main_data" title="Main Data">
    <p align="center">
      <b>
        <%= appointmentBean.getAppointmentTime(intAppointmentID) %>
      </b>
    </p>

    <p>
      <br/>
      <%= appointmentBean.getAppointmentDetails(intAppointmentID) %>
      <br/>
      <a href="#check_off">Check off this appointment</a><br/>
      <a href="http://localhost:8080/pick_appointment.jsp">Back to appointments list</a>
    </p>
  </card>

  <card id="check_off" title="Check Off">
    <!--设置向服务器发送输入数据的选择项 -->
    <do type="accept">
      <go href="/servlet/ProcessCheckOff" method="post">
        <postfield name="check_off_code" value="$check_off_code"/>
      </go>
    </do>

    <p>
      <input name="check_off_code" emptyok="false" maxlength="6"/>
    </p>
```



```
<p>
    <a href="#main_data">Back to appointment data</a>
    <br/>
    <a href="http://localhost:8080/pick_appointment.jsp">Back to appointments list</a>
</p>
</card>

</wml>
<%-- WML 内容结束 --%>
```

11.7 Servlet 技术在 WAP 开发中的应用

WAP 开发中应用 Servlet 技术时要利用专用 WAP 服务器软件，如 Nokia WAP Server 等提供的开放程序接口来编写具体的程序。我们仍以 WML 为例，它应用 Servlet 编程的规则与应用 JSP 时的规则基本相同，我们不再逐条给出解释。通过下面的简单例子，结合 JSP 的编写规则，大家不难了解 Servlet 编写 WML 页面的基本规则：

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class WapServlet extends HttpServlet
{
    protected void service(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, java.io.IOException
    {
        response.setContentType("text/vnd.wap.wml");
        PrintWriter out = response.getWriter();
        xmlHeader(out);
        out.println("<wml>");
        out.println("<card title = \"JavaTest\\\">");
        out.println("<p>Hello! This comes from a servlet!</p>");
        out.println("</card>");
        out.println("</wml>");
    }
}
```




```
}

public void xmlHeader(PrintWriter out)
{
    out.println("<?xml version=\"1.0\"?>");
    out.println("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\" "+
        "\"http://www.wapforum.org/DTD/wml_1.1.xml\">");
}
}
```

下面我们给出一个简单的 Servlet 程序，它可在 WAP 浏览器上显示 “The simple HelloWorld servlet.” 的信息。程序清单如下：

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet
{
    String m_text;

    public void init(ServletConfig config) throws ServletException
    {
        super.init(config);
        m_text = config.getInitParameter("text");
        if (m_text == null)
        {
            m_text = "This is a simple test servlet.";
        }
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException,
        ServletException
    {
        PrintWriter out = response.getWriter();
        out.println("<?xml version=\"1.0\"?>");
        out.println("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML1.1//EN\" \"http://www.wapforum.org/DTD/wml_1.1.xml\">");
    }
}
```

```
out.println("<wml>");
out.println("<card id=\"card1\" title=\"Hello World\">");
out.println("<p>");

out.println(m_text);
out.println("</p>");
out.println("</card>");
out.println("</wml>");
// Remember to close the out object
out.close();
}

public String getServletInfo()
{
    return "The simple HelloWorld servlet.";
}

}
```

11.8 HTML 过滤器和 HTML 页面的转换

HTML 过滤器的功能是将 HTML 页面转换为 WAP 页面。这显然是一个非常好的工具，借助这一工具我们可以把大量已经开发好的基于 HTML 的网站直接转换为 WAP 网站。但是，由于 WAP 浏览器具有内存少、屏幕小的特点，所以 HTML 过滤器的效果并不像我们想像得那么好。下面我们就介绍几个比较有名的 HTML 过滤器。

11.8.1 Wapitout

Wapitout 是一个在线实现 HTML 页面到 WAP 页面转换的过滤器。有关它的详细情况可访问它的站点 <http://www.wapitout.com>。

Wapitout 的使用方法是这样的，在当前 WAP 页面中加入到 Wapitout 过滤器的超链接 http://www.kaufcom.com/wapitoutcom/htmltowap_e.wml，代码为：



```
<a href="http://www.kaufcom.ch/wapitoutcom/htmltowap_e.wml">
    HTML pages on your handy/HTMLtoWAP from Wapitout
</a>
```

这样，页面运行时即可建立该过滤器与当前 WAP 站点的联系，并在浏览器中给出一个输入框。从中输入想要访问的基于 HTML 的站点名称，并按确认键，随后 WAP 浏览器即可显示该站点的网页。当然显示时是按照 WAP 浏览器的格式显示的，Wapitout 过滤器在后台已自动把访问的基于 HTML 的站点转换为 WAP 性质的站点。

另外，我们也可以直接在 WAP 页面中增加一个卡片，专门用于输入想要转换的 HTML 性质的站点，让 Wapitout 过滤器进行转换。该卡片的代码如下：

```
<card title="">           //这里可为该卡片起一个名字
    <p></p>
    <p>
        To convert HTML- to WAP-Sites put the URL in this box:<br/>
        <input type="text" name="adresse" size="18" title="URL-Adresse" value="http://">
        <anchor>
            <go href="http://www.kaufcom.ch/cgi-bin/umwandlerhtml2wml.pl" method="get">
                <postfield name="adresse" value="$(adresse:e)"/>
            </go>
            Display Site
        </anchor>
    <br/>
</p>
</card>
```

11.8.2 TransWap

TransWap 也是一个比较实用的 HTML 过滤器。它是通过在现有的 HTML 页面中加入一对附加的标签来实现 HTML 页面的转换，使得这些页面可被 WAP 设备访问。有关 TransWap 的详细情况可访问它的站点 <http://amaro.g-art.nl/>。

TransWap 要求在 HTML 文件中加入的附加标签内容为：

```
<!-- WAP_START -->
```

this is the part of the HTML-document you want to display

<!-- WAP_END -->

因为其中的标签属于注释标签，不会对 HTML 的浏览器产生任何影响。因此，我们可以把这三行代码写到 HTML 代码中。

但当通过 TransWap 过滤器来访问这些 HTML 页面时，TransWap 就会识别上述标签，并将当前的 HTML 页面转换为 WAP 页面来显示，这样就达到了 WAP 浏览器浏览的目的。

使用 TransWap 过滤器来访问 HTML 页面并使之转换为 WAP 页面的方法很简单，只需在 TransWap 过滤器 URL 地址的后面加上 “/?u=”，再加上要访问的 HTML 页面的 URL 地址即可。其一般格式为：

<http://amaro.g-art.nl/wap/?u=<your homepage>.html>

其中 “<your homepage>.html” 即为想要使用 WAP 浏览器访问的 HTML 页面文件或其 URL 地址。

例如，我们想用 WAP 浏览器访问基于 HTML 的站点 <http://www.myhtml.com>，则输入下述 URL 地址即可：

<http://amaro.g-art.nl/wap/?u=www.myhtml.com/>

再如，若要访问 HTML 页面 <http://www.myhtml.com/index.html>，则可输入以下 URL 地址：

<http://amaro.g-art.nl/wap/?u=www.myhtml.com/index.html>

11.8.3 Coollie

Coollie 过滤器可以将整个基于 HTML 的站点转换成 WAP 站点。操作时用户需要提供想要转换的基于 HTML 的站点，以及转换后存放页面的 WAP 目录。有关 Coollie 过滤器的详细情况可访问它的站点 <http://www.krisn.com/>。Coollie 过滤器是一个应用软件，用户可以到它的站点去下载并把它安装到所用计算机系统中。Coollie 要求用户的计算机中安装有 Java Runtime Environment (JRE) 1.2.2 或其以上版本，才可正常运行。

Coollie 过滤器启动后，屏幕上即出现它的窗口，从中用户可输入想要转换的 HTML 站

点, 然后输入用于保存转换后 WML 页面文件的目录, 另外用户还可以选择 Web 服务器, Coolie 会判断该服务器是否够支持 WAP, 如果支持 WAP 则转换后可得到 wml 格式的文件, 否则就会得到 asp 格式的文件。随后单击“开始”按钮, Coolie 即开始进行转换工作。

完成转换后, 将得到的 WML 页面文件从其存放的目录中复制或上传到 WAP 服务器, 以后用户就可使用 WAP 模拟器或 WAP 客户端设备进行测试、浏览了。

本章小结

本章主要讲解 WAP 编程与开发的高级技术, 涉及 WAP 页面中汉字与图像的使用问题、HTML 页面向 WAP 页面的转换问题, 以及应用 ASP、PHP、Perl、C、JSP、Servlet 等技术开发动态 WAP 页面的问题。本章内容虽然不多, 但涉及的技术却不少, 读者应当具备 ASP、PHP、Perl、C、JSP、Servlet 及 Web 数据库等方面的开发基础。如果没有这些基础, 阅读本章则会感到比较困难, 那样的话大家就应当尽力去补充这方面的知识。

对大多数读者来说, 掌握 ASP、PHP、Perl、C、JSP、Servlet 及 Web 数据库等语言或技术在 WAP 开发中的应用方法, 主要是掌握 WAP 编程语言 WML 与这些语言结合使用时, 书写 WAP 程序的基本规则和具体方法。我们给出的例子比较简单, 仅供学习基本规则之用, 但本书所附光盘中我们提供了一些比较大的、复杂的综合开发实例, 读者可以通过这些实例深刻认识 WAP 编程的高级技术, 开发出复杂的 WAP 应用。



第 12 章 WAP 安全与实现

第 12 章 WAP 安全与实现

与 Internet 上的电子商务和网络传输一样, WAP 网络应用也存在安全性问题。例如, 用户通过 WAP 手机上网, 当通过无线网络发送出自己的个人资料时, 就有可能被人截获、监听或破坏其中的重要数据, 从而会给用户带来一些麻烦甚至重大的损失。为此, 人们做了许多研究工作, 以便实现 WAP 安全。目前, 为无线网络的数据传递提供安全服务的主要工具就是 WAP 的无线传输安全层协议 WTLS(Wireless Transport Layer Security)。这是一种基于安全会话层协议 SSL(Security Socket Layer)的安全传输协议, 主要用于微型浏览器(如手机浏览器)和 WAP 服务器之间, 它能够使用数字凭证(也称为数字证书)在两个实体之间创建一个安全而秘密的通信“管道”, 典型的两个实体就是移动电话和 WAP 服务器。通过 WTLS 连接而进行传输的数据, 如果一旦被篡改或伪造, 那么 WTLS 就会给出警告信息, 提醒用户。

WTLS 使用的数字凭证需要利用数字签名和密钥技术实现, 而它们又涉及数据加密原理和算法。为了说明 WAP 安全的实现原理及方法, 我们先讲解数据加密原理和加密算法, 然后再讲解数字凭证、数字签名、WAP 客户端与服务器端的认证方法等知识。

12.1 数据加密原理与实现方法

无论是网络传输中使用的安全传输协议, 还是安全手段或安全措施, 它们都需要使用加密算法和相应的解密算法。不了解与此相关的加密原理和算法, 就很难深刻地认识各种安全传输协议, 以及实现网络安全所采用的数字凭证、数字签名等技术, 所以我们在本节集中介绍一下。

12.1.1 基本概念

加密方法属于计算机密码学范畴。长久以来,计算机密码学作为一门研究计算机数据加密、解密及其变换的艰深的学科,鲜为普通用户所了解。过去只有间谍及军事人员对加密技术感兴趣,并投入了大量人力、物力和财力进行秘密研究。直到最近十几年,随着计算机网络及通信技术的民用化发展,尤其是商业和金融事务的介入,密码学的研究才得到了前所未有的广泛重视。密码学涉及到很多高深的数学理论,很难在一节内容中进行比较全面的论述。这里我们仅介绍一些基础知识,让读者了解网络信息加密及数字签名、数字凭证的实现原理,并对常用加密方法的优缺点有一些基本认识。

所谓加密,就是把数据信息即明文转换为不可辨识的形式即密文的过程,目的是使不应了解该数据信息的人不能够知道和识别。将密文转变为明文的过程就是解密。加密和解密过程形成加密系统,明文与密文统称为报文。任何加密系统,不论形式如何复杂,实现的算法如何不同,但其基本组成部分是相同的,通常都包括如下 4 个部分:

- (1) 需要加密的报文,也称为明文;
- (2) 加密以后形成的报文,也称为密文;
- (3) 加密、解密的装置或算法;
- (4) 用于加密和解密的钥匙,称为密钥。密钥可以是数字、词汇或者语句。

报文加密后,发送方就要将密文通过通信渠道传输给接收方。传输过程中,即密文在通信渠道传输过程中是不安全的,可能被非法用户即第三方截取和窃听,但由于是密文,只要第三方没有密钥,只能得到一些无法理解其真实意义的密文信息,从而达到保密的目的。整个过程如图 12.1 所示。

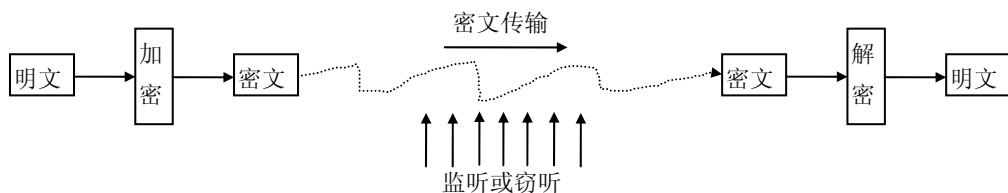


图 12.1 加密与解密过程

长久以来,人们发明了各种各样的加密方法,为便于研究,通常把这些方法分为传统加密方法和现代加密方法两大类。前者的共同特点是采用单钥技术,即加密和解密过程中使用同一密钥,所以它也称为对称式加密方法;而后者的共同特点是采用双钥技术,也就



是加密和解密过程中使用两个不同的密钥，它也称为非对称式加密方法。下面就简单介绍一下这些加密方法。

12.1.2 基于单钥技术的传统加密方法

这类方法主要包括代码加密法、替换加密法、变位加密法和一次性密码簿加密法等。

(1) 代码加密法。通信双方使用预先设定的一组代码表达特定的意义，而实现的一种最简单的加密方法。代码可以是日常词汇、专用名词，也可以是某些特殊用语。例如：

密文：姥姥家的黄狗三天后下崽。

明文：县城鬼子三天后出城扫荡。

这种方法简单好用，但通常一次只能传送一组预先约定的信息，而且重复使用时是不安全的，因为那样的话窃密者会逐渐明白代码含义。

(2) 替换加密法。这种方法是制定一种规则，将明文中的每个字母或每组字母替换成另一个或一组字母。例如，下面的这组字母对应关系就构成了一个替换加密器：

明文字母：A B C D E F……

密文字母：K U P S W B……

虽然说替换加密法比代码加密法应用的范围要广，但使用得多了，窃密者就可以从多次搜集的密文中发现其中的规律，破解加密方法。

(3) 变位加密法。与前两种加密方法不同，变位加密法不隐藏原来明文的字符，而是将字符重新排序。比如，加密方首先选择一个用数字表示的密钥，写成一行，然后把明文逐行写在数字下。按照密钥中数字指示的顺序，将原文重新抄写，就形成密文。例如：

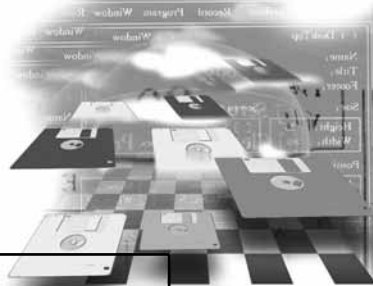
密钥：6835490271

明文：小赵拿走黑皮包交给李

密文：包李交拿黑走小给赵皮

(4) 一次性密码簿加密法。这种方法要先制定出一个密码簿，该簿每一页都是不同的代码表。加密时，使用一页上的代码加密一些词，用后撕掉或烧毁该页；然后再用另一页上的代码加密另一些词，直到全部的明文都加密成为密文。破译密文的唯一办法就是获得一份相同的密码簿。

计算机出现以后，密码簿就无需使用纸张而使用计算机和一系列数字来制作。加密时，



根据密码簿里的数字对报文中的字母进行移位操作或进行按位的异或计算，以加密报文。解密时，接收方需要根据持有的密码簿，将密文的字母反向移位，或再次作异或计算，以求出明文。

数论中的“异或”规则是这样的： $1^1=0$ ， $0^0=0$ ； $1^0=1$ ， $0^1=1$ 。下面就是一个按位进行异或计算的加密和解密实例：

加密过程中明文与密码按位异或计算，求出密文：

明文：101101011011

密码：011010101001

密文：110111110010

解密过程中密文与密码按位异或计算，求出明文：

密文：110111110010

密码：011010101001

明文：101101011011

顾名思义，一次性密码簿只能使用一次，以保证信息加密的安全性。但由于解密时需要密码簿，所以想要加密一段报文，发送方必须首先安全地护送密码簿到接受方(这一过程常称为“密钥分发”过程)。如果双方相隔较远，如从美国五角大楼到英国中央情报局，则使用一次性密码簿的代价是很大的。这也是限制这种加密方法实用化和推广的最大障碍，因为既然有能力把密码簿安全地护送到接受方，那为什么不直接把报文本身安全地护送到目的地呢？

正因为传统加密方法在这方面的局限性，人们又想出了很多算法来加强和改进这些方法。下面我们就介绍几个比较著名的方法。

12.1.3 改进的传统加密方法

我们首先需要说明数据块和数据流加密的概念。数据块加密是指把数据划分为某一特定长度的数据块，再分别进行加密。数据块之间的加密是相互独立的，因此，如果内容相同的数据块重复出现，密文也会呈现出某种规律性，从而会降低破密的难度。数据流加密是指使用加密后的密文前面的部分，来参与报文后面部分的加密。这种方法的好处是数据块之间的加密不再独立，即使有相同的数据重复出现，密文也不会呈现出明显的规律性，从而提高破译的难度。改进的传统加密方法便是应用了这种思想，这类方法常划为使用传



统加密技术的现代加密方法。

(1) 数据加密标准 DES。DES(Data Encryption Standard)是美国政府 1977 年采用的加密标准,最初是由 IBM 公司在 70 年代初期开发的。美国政府在 1981 年又将 DES 进一步规定为 ANSI 标准。

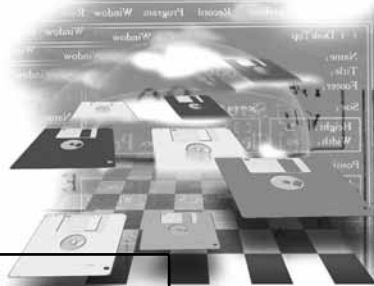
DES 是一个对称密钥系统,加密和解密使用相同的密钥。它通常选取一个 64 位(bit)的数据块,使用 56 位的密钥,在内部实现多次替换和变位操作来达到加密的目的。DES 有 ECB, CBC 和 CFB 三种工作模式,其中 ECB 采用的是数据块加密模式, CBC 与 CFB 采用的是数据流加密模式。

作为第一个公开的新式加密方法,DES的影响非常大。后来提出的许多密码方法都汲取了DES的思想和技术。但是,DES的缺点是它采用的密钥太短,只有 56 位,也就是说,所有可能的密钥只有 2^{56} 个。采用一些计算网络,若每秒钟测试 5 亿个密钥,则在 4 小时以内便可把所有可能的密钥都测试一遍。因此,随着计算机性能的提高,DES的破解难度已经降低,不太实用了。

(2) 三层 DES(Triple-DES)。这种方法是 DES 的改进加密算法,它使用两把密钥对报文作三次 DES 加密,效果相当于将 DES 密钥的长度加倍。三层 DES 克服了 DES 的显著缺点,即其 56 位的短密钥。本来,三层 DES 是通过三次使用 DES 算法来对数据进行编码加密,在每一层上都使用不同的密钥,这样就可以用一个 $3 \times 56 = 168$ 位的密钥进行加密;但许多密码设计者认为 168 位的密钥已经超过实际需要了,所以便在第一层和第三层中使用相同的密钥,产生一个有效的 112 位的密钥长度。之所以没有直接采用两层 DES,是因为第二层 DES 不是十分安全,它对一种称为“中间可遇”的密码分析攻击形式来说是极为脆弱的,所以还是采用了三层 DES 操作。

(3) RC2 和 RC4。RC 指 Rivest Code,它是以发明人美国麻省理工学院的 Ron Rivest 教授的姓氏命名的,由 RSADSI 公司发行,是不公开的专有算法。RC2 和 RC4 使用可变长度(1 至 1024 位)的密钥,实现不同级别的保密性。RC2 采用的是数据块加密算法,RC4 采用的是数据流加密算法。由于它们的具体算法不公开,所以没有人知道它们的可靠性到底能达到何种程度。有名的 Netscape 浏览器从 2.0 版就使用了 RC2 和 RC4 的加密算法。

(4) 数字摘要(Digital Digest)。该加密方法也是由 Ron Rivest 设计的,也被称为安全 Hash 编码法 SHA(Secure Hash Algorithm)或 MD5(MD Standards for Message Digest)。该编码法采用



单向Hash函数¹将需加密的明文“摘要”成一串 128bit(位)的密文,这一串密文也称为数字指纹(Finger Print),它具有固定的长度。而且不同的明文摘要成密文时,其指纹结果也是不同的,而相同明文的摘要必定相同。这样,这串摘要便可以成为验证明文是否是“真身”的“指纹”了。SHA其实就是RC方法的一种实现。

(5) 国际数据加密算法 IDEA。IDEA(International Data Encryption Algorithm)是 1990 年瑞士的 James Massey, Xuejia Lai 等人发表的一个数据块加密算法。该算法使用 128 位的密钥,能够有效地消除试图穷尽搜索密钥的可能攻击。IDEA 看起来是一个较“强”的算法,但由于出现没多久,所以目前暂还没人知道它是否有漏洞。

(6) 基于硬件的加密。为克服软件加密算法在容易复制、容易尝试方面的不足,人们又开发了基于硬件的加密算法。如美国国家安全局为使用 Clipper 芯片,就秘密开发了一个民用加密算法 SkipJack,采用 80 位的密钥,使得穷尽搜索密钥不可行。而且由于在 Clipper 芯片的硬件中人为地加进了一些“机关”设置,增加了破解难度。

虽然说传统的加密方法在军事、谍报、金融和其他商业领域曾一度得到了广泛应用,但它在密钥分发方面存在一些弊端,主要有下面 3 个方面:

其一,接收方必须有密钥才能解密,为此就需分发密钥,而安全送达密钥的代价往往很大。

其二,多人通信时密钥的组合数量往往很大,使得密钥选取和分发变得十分困难;例如,3 个人两两通信时总共只需 3 把密钥,但若 6 人两两通信则总共需要 15 把密钥, n 个人两两通信共需要密钥数为 $n(n-1)/2$ 把;如果一个 100 多人的团体内部进行两两通信,则需要安全地分发近 5000 把密钥,代价实在太太。

其三,当通信人增多、密钥增多时,密钥的管理非常困难;因为密钥的管理人员和传送人员中,如果有人受贿或背叛,则密钥就可能被泄露,从而失去保密的意义。

这些弊端说明传统加密方法有很大的局限性,人们必须寻找其他的方法,实现更方便实用的加密目的,来保护网络通信信息和数据。公共密钥加密方法便是人们在传统加密方法的基础上,提出的一种更为有效的加密方法。

¹ 有关Hash函数和IDEA的内容请参见:卢开澄. 计算机密码学. 北京:清华大学出版社, 1990。



12.1.4 基于双钥技术的现代加密方法

我们先讲解一下双钥技术的工作原理，然后再介绍几个著名的加密方法。

工作原理分析

双钥技术就是公共密钥加密 PKE(Public Key Encryption)技术，它使用两把密钥，一把公共密钥(Public Key)和一把专用密钥(Private Key)，前者用于加密，后者用于解密。这种方法也称为“非对称式”加密方法，它解决了传统加密方法的根本性问题，极大地简化了密钥分发的工程量。它与传统加密方法相结合，还可以进一步增强传统加密方法的可靠性。更为突出的是，利用公共密钥加密技术可以实现数字签名。

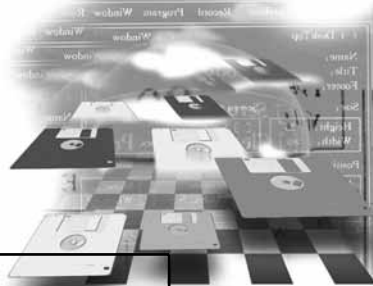
为说明公共密钥技术的工作原理，我们还得从传统加密方法的工作原理说起。

传统加密方法的工作过程包括如下几部分：

- (1) 设置一个保险箱，并装置一把嵌入箱子的暗锁，这种锁只能使用钥匙才能锁上，再准备 2 把相同的钥匙。这对应于加密方法，钥匙对应于密钥。
- (2) 发信方和收信方必须各自持有 1 把开锁的钥匙。这对应于密钥分发过程。
- (3) 发信方将通信文件放入保险箱，并使用自己持有的钥匙把锁锁起来。这对应于加密过程。
- (4) 运送保险箱。这对应于信息传输过程。
- (5) 保险箱送到目的地后，收信人用自己持有的钥匙打开锁，取出通信文件。这对应于解密过程。

全部过程的根本困难是第(2)步，即通信双方必须都获得 1 把统一的钥匙，前文已经述及，这一步实现起来要付出很大代价。如果我们仔细分析一下，可以发现，发信人真正需要的仅仅是一个能装秘密文件并能锁上的保险箱，没有必要也持有 1 把钥匙。之所以要有钥匙，是因为使用了嵌入箱子的暗锁。显然，只要不使用这种锁，而使用一种不用钥匙也能锁上的锁，则发信人就可以在没有钥匙的情况下，锁住保险箱。基于这种思想，可以按如下过程进行加密：

- (1) 设置一个保险箱，并设置一把不需使用钥匙就能锁上的锁，再准备 1 把钥匙。这对应于加密方法，钥匙对应于专用密钥。
- (2) 收信方持有这把开锁的钥匙，同时准备该钥匙可以开的锁，锁可以不止有 1 把，然



后把锁公开发放出去。这里锁对应于公共钥匙，这个过程对应于公共密钥分发过程。

(3) 任何人想与收信方通信，只需将通信文件放入保险箱，并使用收信方承认的锁锁起来。这对应于加密过程。

(4) 运送保险箱。这对应于信息传输过程。

(5) 保险箱达到目的地后，收信人用自己持有的唯一的钥匙打开锁，取出通信文件。这对应于解密过程。

由于可以开锁的钥匙只有一把，而且掌握在收信人手里。因此可以确信除收信人本人外，没有任何人能够打开锁着的保险箱并偷阅其中通信文件的内容，发信人对此也不例外。更为重要的是，密钥分发的难题也不复存在了，而是代之以锁的分发问题。分发锁是无须保密的，这无疑使以前的艰难工作简单了许多。这就是公共密钥加密技术的工作原理。

公共密钥加密系统中，收信人首先生成在数学上相互关联、但又不相同的两把钥匙，一把公共密钥用于加密，另一把专用密钥用于解密，这一过程称为密钥配制过程。其中公共密钥相当于例子中不需使用钥匙就能锁上的锁，用于以后通信的加密；另一把专用密钥相当于例中提到的那把开锁的唯一的钥匙，用于通信的解密。收信人将唯一的专用密钥掌握和保存起来，把公共密钥通过各种方式公布出去，让想与收信人通信的人都能够得到。这个过程就是公共密钥的分发过程。发信人使用收信人的公共密钥对通信文件进行加密，加密后的密文发信人自己也无法解开，这相当于把信件十分可靠地锁在保险箱里。收信人在收到密文以后，用自己的专用密钥解开密文获得明文信息。

公共密钥加密系统的优点

与传统加密方法相比，公共密钥加密系统具有 3 方面比较突出的优点：

(1) 用户可以把用于加密的密钥，公开地分发给任何需要的其他用户。谁都可以使用这把公共的加密密钥与该用户秘密通信。除了持有解密密钥的收件方用户外，没有人能够解开密文。这样，传统加密方法中令人头痛的、代价沉重的密钥分发问题就转变为一个性质完全不同的“公共密钥分发”的问题。

(2) 公共密钥加密系统允许用户事先把公共密钥发表或刊登出来。譬如，用户可以把它和电话号码、产品说明等一起刊登出来，让任何人都可以查找并使用到。这使得公共密钥应用的范围不再局限于信息加密，还可以应用于身份鉴别、权限区分等各种领域。例如，大家熟知的各种应用软件，如 Windows 95/98 等系统安装时需要的产品序列号，其实就是公共密钥，它通常印在产品授权书的封面或封底上，供安装时鉴别用户的授权身份。

(3) 公共密钥加密不仅改进了传统加密方法,而且还提供了传统加密方法所不具备的应用,即是数字签名的公开鉴定系统。有关数字签名的工作原理与过程我们将在后面介绍。

自从 1976 年第一个正式的公共密钥加密算法提出,又有几个算法被相继提出。下面我们就简要介绍几个主要的公共密钥加密算法。

Ralph Merkle 猜谜法

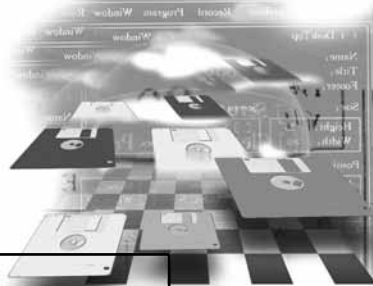
虽然这个方法并没有付诸实施,但它是第一个认真思考公共密钥加密问题的算法。该算法于 1974 年由美国加州大学伯克利分校的学生 Ralph Merkle 提出,核心是做 100 万个猜谜题,每道谜题藏有一个密码,求解一道谜题大约需要 2 分钟时间。该算法的工作流程是这样的:甲方先随机地产生 100 万把密钥,并藏在 100 万道谜题里,然后把这 100 万道谜题发送给乙方;乙方收到后,随意挑选并求解一道谜题,取出其中的那把密钥,再用该密钥对一个双方事先统一的报文加密,这段报文可以任意,且无需保密;加密后,把密文发送给甲方;甲方并不知道乙方选择了哪把密钥,于是使用自己保存的 100 万把密钥逐一进行尝试解密;其中必有一个是可行的,这把密钥不为其他人所知,所以可以作为以后通信使用的密钥。

如果传输过程中被人窃取,那么窃取者可能窃取到甲方发出的 100 万道谜题,也可能窃取到乙方返回的密文。但要想知道乙方选中了哪把密钥,窃取者只能去求解 100 万道谜题,由于解开一个谜题需要 2 分钟时间,按 50%的解出概率计算,窃取者平均需要 23 个月的时间才能找到答案。不过那时通信双方早已结束了通信,完成了秘密任务或改换了密码,窃取者的工作已经没有什么大意义了。

Ralph Merkle 方法有一个弊端,那就是收发双方之间的通信系统必须能够在合理的时间内存送 100 万道谜题,而且甲方应拥有快速的计算机以产生 100 万道谜题和尝试 100 万次搜索,找出乙方选择的密码。如果计算机性能不佳,显然这种方法就不太实用。

Diffie-Hellman 指数密钥交换加密算法

该算法于 1976 年由美国斯坦福大学的 Whitfield Diffie 和 Martin Hellman 提出,利用了离散指数易求而反函数难求的特性来设计加密系统,是专门为双方主动参与的通信过程设计的。该算法非常简捷。它让通信双方共同参与一个数学运算过程,并统一一把用于以后加密的密钥。即使监听者能窃取到全部交换信息,但由于没有参与运算,他也无法获得最终的密钥。通信双方首先各自挑选一个秘密的数,然后交换一些由此数导出的信息;接下



来利用这些信息,通过离散指数和质数求余等运算就可以进一步推导出一把统一的密钥,用于加密以后的通信。该密钥交换加密算法的工作过程如下:

(1) 通信双方统一两个数 D 和 H 。 D 与 H 无须对外保密。

(2) 双方各自选择一个秘密的数 X ,并把包括 D , H 和 X 的计算结果 Y 发送给对方。例如:假设甲方选中了 X_1 ,则发出了 Y_1 ;乙方选中了 X_2 ,则发出了 Y_2 。

(3) 根据算法,双方各自使用自己选择的 X 和收到的 Y ,计算出一个统一的数字 K 。 K 就是双方进一步通信的会话密钥。具体地讲, K 可以从 (X_1, Y_2) 或 (X_2, Y_1) 中导出,但却不能从 (Y_1, Y_2) 中得到。这一点的重要意义在于,窃听者虽然可以得到 D , H , Y_1 和 Y_2 ,甚至密文,但由于不知道 X_1 和 X_2 的值,因此无法获得导出正确的会话密钥 K ,从而也就无法破解密文。

(4) 以后通信时,双方使用 K 进行加密和解密。

Diffie-Hellman 算法的弊端在于它需要收发双方必须同时参与密码的生成过程,所以它不适用于电子函件的加密,因为电子函件在收信人不在场时也应能够发送过去。

RSA 加密算法

该算法于 1977 年由美国麻省理工学院 MIT(Massachusetts Institute of Technology)的 Ronal Rivest, Adi Shamir 和 Len Adleman 三位年轻教授提出,并以三人的姓氏 Rivest, Shamir 和 Adleman 命名为 RSA 算法。该算法利用了数论领域的一个事实,那就是虽然把两个大质数相乘生成一个合数是件十分容易的事情,但要把一个合数分解为两个质数却十分困难。合数分解问题目前仍然是数学领域尚未解决的一大难题,至今没有任何高效的分解方法。与 Diffie-Hellman 算法相比, RSA 算法具有明显的优越性,因为它无须收发双方同时参与加密过程,且非常适用于电子函件系统的加密。

RSA 算法可以表述如下:

(1) 密钥配制。假设 m 是想要传送的报文,现任选两个很大的质数 p 与 q ,使得:

$$n = p \cdot q > m \quad (12-1);$$

选择正整数 e ,使得 e 与 $(p-1)(q-1)$ 互质;这里 $(p-1)(q-1)$ 表示二者相乘。再利用辗转相除法,求得 d ,使得:

$$(e \cdot d) \bmod (p-1)(q-1) = 1 \quad (12-2);$$

其中 $x \bmod y$ 是整数求余运算,其结果是 x 整除以 y 后剩余的余数,如 $5 \bmod 3 = 2$ 。

这样得:

(e, n) , 是用于加密的公共密钥, 可以公开出去; 以及

(d, n) , 是用于解密的专用密钥, 必须保密。

(2) 加密过程。使用 (e, n) 对明文 m 进行加密, 算法为:

$$c = m^e \bmod n \quad (12-3);$$

这里的 c 即是 m 加密后的密文。

(3) 解密过程。使用 (d, n) 对密文 c 进行解密, 算法为:

$$m = c^d \bmod n \quad (12-4);$$

求得的 m 即为对应于密文 c 的明文。

RSA 算法实现起来十分简捷, 据说英国的一位程序员只用了 3 行 Perl 程序便实现了加密和解密运算。

RSA 算法建立在正整数求余运算基础之上, 同时还保持了指数运算的性质, 这一点我们不难证明。例如:

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n \quad (12-5);$$

$$m^{(a+b)} \bmod n = (m^a \bmod n)(m^b \bmod n) \bmod n \quad (12-6)。$$

RSA 公共密钥加密算法的核心是欧拉(Euler)函数 ψ 。对于正整数 n , $\psi(n)$ 定义为小于 n 且与 n 互质的正整数的个数。例如 $\psi(6) = 2$, 这是因为小于 6 且与 6 互质的数有 1 和 5 共两个数; 再如 $\psi(7) = 6$, 这是因为互质数有 1, 2, 3, 5, 6 共 6 个。

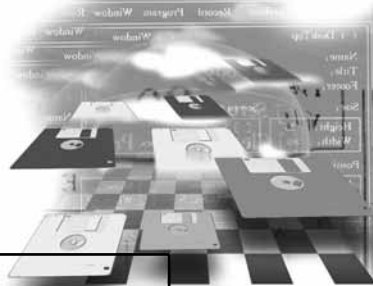
欧拉在公元前 300 多年就发现了 ψ 函数的一个十分有趣的性质, 那就是对于任意小于 n 且与 n 互质的正整数 m , 总有 $m^{\psi(n)} \bmod n = 1$ 。例如, $5^{\psi(6)} \bmod 6 = 5^2 \bmod 6 = 25 \bmod 6 = 1$ 。也就是说, 在对 n 求余的运算下, $\psi(n)$ 指数具有周期性。

当 n 很小时, 计算 $\psi(n)$ 并不难, 使用穷举法即可求出; 但当 n 很大时, 计算 $\psi(n)$ 就十分困难了, 其运算量与判断 n 是否为质数的情况相当。不过在特殊情况下, 利用 ψ 函数的两个性质, 可以极大地减少运算量。

性质 1: 如果 p 是质数, 则 $\psi(p) = (p-1)$ 。

性质 2: 如果 p 与 q 均为质数, 则 $\psi(p \cdot q) = \psi(p) \cdot \psi(q) = (p-1)(q-1)$ 。

RSA 算法正是注意到这两条性质来设计公共密钥加密系统的, p 与 q 的乘积 n 可以作为公共密钥公布出来, 而 n 的因子 p 和 q 则包含在专用密钥中, 可以用来解密。如果解密需要用到 $\psi(n)$, 收信方由于知道因子 p 和 q , 可以方便地算出 $\psi(n) = (p-1)(q-1)$ 。如果窃



听者窃得了 n ，但由于不知道它的因子 p 与 q ，则很难求出 $\psi(n)$ 。这时，窃听者要么强行算出 $\psi(n)$ ，要么对 n 进行因数分解求得 p 与 q 。然而，我们知道，在大数范围内作合数分解是十分困难的，因此窃密者很难成功。

有了关于 ψ 函数的认识，我们再来分析 RSA 算法的工作原理：

(1) 密钥配制。设 m 是要加密的信息，任选两个大质数 p 与 q ，使得 $n = p \cdot q > m$ ；选择正整数 e ，使得 e 与 $\psi(n) = (p-1)(q-1)$ 互质。

利用辗转相除法，计算 d ，使得 $ed \bmod \psi(n) = (e \cdot d) \bmod (p-1)(q-1) = 1$ ，即 $ed = k\psi(n) + 1$ ，其中 k 为某一正整数。

公共密钥为 (e, n) ，其中没有包含任何有关 n 的因子 p 和 q 的信息。

专用密钥为 (d, n) ，其中 d 隐含有因子 p 和 q 的信息。

(2) 加密过程。使用公式(12-3)对明文 m 进行加密，得密文 c 。

(3) 解密过程。使用 (d, n) 对密文 c 进行解密，计算过程为：

$$\begin{aligned} c^d \bmod n &= (m^e \bmod n)^d \bmod n \\ &= m^{ed} \bmod n \\ &= m^{(k\psi(n) + 1)} \bmod n \\ &= (m^{k\psi(n)} \bmod n) \cdot (m \bmod n) \\ &= m \end{aligned}$$

m 即为从密文 c 中恢复出来的明文。

例如，假设我们需要加密的明文代码信息为 $m = 14$ ，则：

选择 $e = 3$ ， $p = 5$ ， $q = 11$ ；

计算出 $n = p \cdot q = 55$ ， $(p-1)(q-1) = 40$ ， $d = 27$ ；

可以验证： $(e \cdot d) \bmod (p-1)(q-1) = 81 \bmod 40 = 1$ ；

加密： $c = m^e \bmod n = 14^3 \bmod 55 = 49$ ；

解密： $m = c^d \bmod n = 49^{27} \bmod 55 = 14$ 。

关于 RSA 算法，还有几点需要进一步说明：

(1) 之所以要求 e 与 $(p-1)(q-1)$ 互质，是为了保证 $ed \bmod (p-1)(q-1)$ 有解。

(2) 实际操作时，通常先选定 e ，再找出并确定质数 p 和 q ，使得计算出 d 后它们能满足公式(12-3)。常用的 e 有 3 和 65537，这两个数都是费马序列中的数。费马序列是以 17 世纪法国数学家费马命名的序列。



(3) 破密者主要通过将 n 分解成 $p \cdot q$ 的办法来解密, 不过目前还没有办法证明这是唯一的办法, 也可能有更有效的方法, 因为因数分解问题毕竟是一个不断发展的领域, 自从 RSA 算法发明以来, 人们已经发现了不少有效的因数分解方法, 在一定程度上降低了破译 RSA 算法的难度, 但至今还没有出现动摇 RSA 算法根基的方法。

(4) 在 RSA 算法中, n 的长度是控制该算法可靠性的重要因素。目前 129 位、甚至 155 位的 RSA 加密勉强可解, 但目前大多数加密程序均采用 231、308 甚至 616 位的 RSA 算法, 因此 RSA 加密还是相当安全的。

据专家测算, 攻破 512 位密钥 RSA 算法大约需要 8 个月时间; 而一个 768 位密钥 RSA 算法在 2004 年之前无法攻破。现在, 在技术上还无法预测攻破具有 2048 位密钥的 RSA 加密算法需要多少时间。美国 Lotus 公司悬赏 1 亿美元, 奖励能破译其 Domino 产品中 1024 位密钥的 RSA 算法的人。从这个意义上说, 遵照 SET 协议开发的电子商务系统是绝对安全的。

Merkle-Hellman 背包算法

该算法是根据数学上的背包问题设计的。背包问题是一个最优化问题, 即对一个给定空间或负重的背包和许多大小不一的物体, 哪些物体放入背包才能使得浪费的背包空间或负重最小? 在背包很小和物体数目较少时, 这个问题还比较容易解决; 但当背包很大且有很多个物体时, 问题的求解就十分困难。通常, 这个问题会有一个或者多个解, 也有可能根本没有解。

1977 年, Merkle 与 Hellman 合作设计了使用背包问题实现信息加密的方法。其工作原理是: 假定甲想加密, 则先产生一个较易求解的背包问题, 并用它的解作为专用密钥; 然后从这个问题出发, 生成另一个难解的背包问题, 并作为公共密钥。如果乙想向甲发送报文, 乙就可以使用难解的背包问题对报文进行加密, 由于这个问题十分难解, 所以一般没有人能够破译密文; 甲收到密文后, 可以使用易解的专用密钥解密。

该算法提出以后, 经过多年的探讨和研究, 最终发现了它的一个致命错误, 使之失去了任何保密的实用价值。



12.2 实现 WAP 安全的一般方法

利用前面讲解的加密方法,尤其是基于双钥技术的现代加密方法,我们针对网络安全可以实现多种具体的手段及方法,如数字签名、数字时间戳、数字凭证及认证中心等,而且这些方法和手段常常结合在一起使用,长短互补,从而构成了网上安全防范的实用体系。WAP 的无线网络在寻求安全技术时,也借用了普通 Internet 网络的安全手段和方法,并将它们进一步调整和优化,以适应无线网络的特殊要求。

12.2.1 数字签名

上一节介绍的密钥对与数字摘要也可以结合起来使用,数字签名就是它们结合的结果。下面我们就讲解一下数字签名是怎么一回事。

日常生活中,我们常把在书面文件上签名作为确认文件的一种手段。签名的作用有两点,一是因为自己的签名难以否认,从而可以确认文件已经签署的事实,即保证文件的不可否认性;二是因为签名不易仿冒,从而可以确定文件是真的这一事实,即保证文件的真实性。网络安全防范中的数字签名(Digital Signature)与书面文件签名有相同之处,它也能确认两点:其一,信息是由签名者发送的;其二,信息自签发后到收到为止未曾作过任何修改。

这样,数字签名就可以用来防止网上电子信息因容易修改而被人破坏或作伪,或冒用别人名义发送信息,或者,在发出或收到信件后又加以否认等情况发生。

网上的数字签名并不是使用“手书签名”类型的图形标志,因为这样也是极易复制的。为此,数字签名另辟蹊径,采用了双重加密的方法来实现加密文件的防伪与防赖。其原理及处理过程如下:

- (1) 使用 SHA 编码将发送文件加密产生 128bit(位)的数字摘要;
- (2) 发送方用自己的专用密钥对摘要再加密,这样就形成了数字签名;
- (3) 将原文和加密的摘要同时传给对方;
- (4) 接收方用发送方的公共密钥对摘要解密,同时对收到的文件用 SHA 编码加密产生又一摘要;
- (5) 将解密后的摘要和收到的文件在接收方重新加密产生的摘要相互对比,如果两者一

致，则说明传送过程中信息没有破坏和篡改。否则，则说明信息已失去其安全性及保密性。

图 12.2 说明了数字签名的过程。

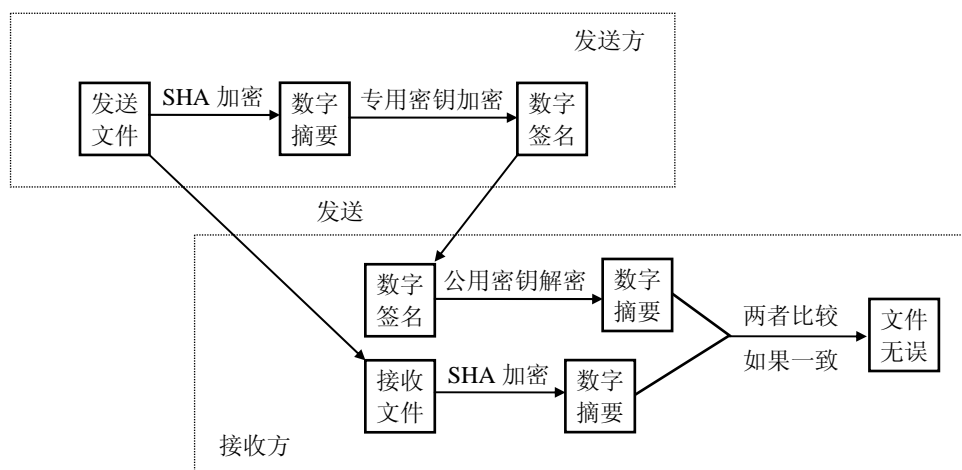


图 12.2 数字签名过程

12.2.2 数字时间戳

时间是网络传输中的一项非常重要的信息。在书面合同中，文件签署的日期和签名一样都是十分重要的防止文件伪造和篡改的关键性证据。

在网络安全防范中，同样需要对传输资料文件的日期和时间信息采取安全措施，这是通过数字时间戳服务 DTS(Digital Time-stamp Service)实现的，它可以对电子文件提供发表时间的安全保护。该服务是 Internet 网上的安全服务项目，由专门的网络服务机构提供。


时间戳(Time-stamp)是一个经过加密而形成的凭证文档，由 3 部分组成：

- (1) 需要加载时间戳的文件摘要；
- (2) DTS 收到文件的日期和时间；
- (3) DTS 的数字签名。

时间戳的产生过程是这样的：用户首先将需要加时间戳的文件用 Hash 编码加密形成摘要，然后将该摘要发送到 DTS，DTS 在加入该文件摘要的收到日期和时间信息后再对该文件加密，即进行数字签名，最后送回用户。



此外, Bellcore 创造了 DTS 的另一种处理方法, 过程如下: 加密时先将摘要信息归并到二叉树的数据结构, 然后再将二叉树的根值发表在报纸上, 这样就能更为有效地为文件发表时间提供辅助证据。

 需要注意的是, 书面签署文件的时间是由签署人自己写上的, 而数字时间戳则不然, 它是由认证服务机构 DTS 来加的, 以 DTS 收到该文件的时间为依据。因此, 时间戳也可以作为各种科技文献的发表证明。

12.2.3 数字凭证

数字凭证(Digital Certificate 或 Digital ID)又称为数字证书或数字标识, 是 Internet 上使用电子手段证实用户身份和用户访问网络资源权限的一种安全防范手段。网上活动的双方当事人如果都出示了各自的数字凭证, 并且用它们进行各种操作, 那么双方都不必为对方身份的真伪而担心。也就是说, 数字凭证保证了信息处理参与各方的身份真实性。

数字凭证可广泛应用于电子函件、网上交易、群件、电子基金转移等各种网络信息环境。

数字凭证的内部格式

数字凭证的内部格式是由 CCITT X.509 国际标准规定的, 包含以下 6 项内容:

- (1) 凭证拥有者的姓名;
- (2) 凭证拥有者的公共密钥;
- (3) 公共密钥的有效期;
- (4) 颁发数字凭证的单位;
- (5) 数字凭证的序列号(Serial Number);
- (6) 颁发数字凭证单位的数字签名。

数字凭证的一般类型

普通 Internet 网络的数字凭证主要有以下 3 种类型:

(1) 个人凭证(Personal Digital ID)。这种类型的数字凭证仅仅可以为单个用户提供凭证, 以帮助用户在网上证明自己的身份, 并从事各种网络活动, 如收发 E-mail、网上购物等。个人身份的数字凭证通常是安装在客户端的 Internet 浏览器内, 并通过安全的多用途网际邮



件扩充协议 S/MIME(Secure/Multipurpose Internet Mail Extension protocol)控制操作。

(2) 企业服务器凭证(Server ID)。这是为网上的企业的 Web 服务器提供凭证的数字凭证类型,拥有 Web 服务器的企业可以使用具有凭证的 Internet 网络站点进行各种网络活动。有凭证的 Web 服务器会自动地将其与客户端通信的信息进行加密。

(3) 软件开发者凭证(Developer ID)。这是为 Internet 中被下载的软件提供的凭证,主要用于与 Microsoft 公司 Authenticode 技术(指“合法化软件”)结合的软件,以使用户在下载软件时能够获得需要的信息。

WAP 系统中的数字凭证

WAP 系统中使用的数字凭证采用了普通 Internet 网络数字凭证的基本思想。因为 WAP 实现安全防范的专门协议就是无线传输层安全协议 WTLS(Wireless Transport Layer Security),所以 WAP 针对 WTLS 用户提供了两种数字凭证,另外还针对 WAP 网关制定的数字凭证。

(1) WTLS 服务器数字凭证。这是为 WAP 服务器提供的数字凭证,可以用来验证 WTLS 服务器到 WTLS 用户(如手机等手持设备)的通信,同时提供了一个通过密钥加密的客户服务对话机制。WTLS 服务器使用的数字凭证和 SSL 服务器使用的凭证有些类似,不同的地方在于,SSL 的数字凭证属于 X.509 类型,相对来说规模比较大(即数字凭证的电子文件规模),而 WTLS 使用的是一个小型化的数字凭证,与 X.509 型的凭证很像,但更加简单小巧。另外,WTLS 服务器的认证是强制性的,而 SSL 服务器的认证是可选的。

(2) WTLS 客户端数字凭证。该数字凭证用于验证一个 WTLS 用户到服务器的通信和对话。它可以是 X.509 型的凭证,也可以是小型化的凭证。

WAP 系统中也定义了一个基于双密钥技术的加密和解密功能,能使 WAP 客户端对某一事务处理进行数字签名。但它不能提供权威的凭证,因为它是为那些不需要名誉保证的用户来使用的。

(3) 网关数字凭证。一个无线网关通常就是一个网络服务的提供者,能给无线环境和有线 Internet 之间提供连接。这种情况下,网关担当的角色就如同是内容的传输者(图 12.3),能够将 WAP/WTLS 转换成 HTTP/SSL,等等。出于安全考虑,WAP 服务和网关提供商合作,还推出了针对网关的数字凭证,以便实现网关的安全认证和防范。

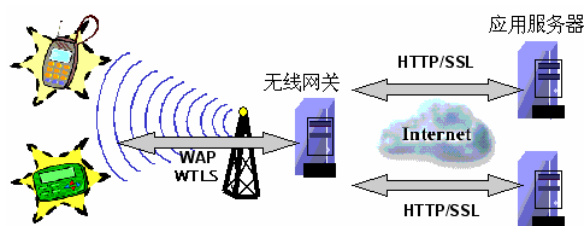


图 12.3 WAP 网关的主要职责

微型凭证

WAP 服务器、客户端及网关使用的数字凭证，大都属于小型化的凭证，我们称这种凭证为微型凭证(Mini-Certificate)。由于这些凭证需要在无线网络中传输，需要在具有有限资源的无线用户端进行处理，所以它的格式非常重要，采用普通有线网络的数字凭证格式显然不能满足要求，所以针对无线网络的传输特点，以及手机等无线设备的内存小、屏幕小、速度慢的特点，数字凭证的提供商优化了传统的凭证格式，在密钥长度、凭证序列号、数字签名大小以及密钥有效期等方面都作了调整，使得微型凭证既能满足实际需求，又能有效抵挡攻击、破解和伪证等。

12.2.4 认证中心

网络安全防范中，无论是数字时间戳服务(DTS)还是数字凭证(Digital ID)的发放，仅靠参与双方是不能圆满完成的，事实上必须需要有一个具有权威性和公正性的第三方(Third Party)来完成。认证中心 CA(Certificatzon Authority)就是承担网络安全认证服务、签发数字证书，并能确认用户身份的服务机构。认证中心通常是企业性的网络服务机构，主要任务是受理数字凭证的申请、签发及对数字凭证的管理，并依据认证实施条例 CPS(Certification Practice Statement)实施服务操作。

主要认证中心

目前在全球处于领导地位的认证中心是美国的 VeriSign 公司。该公司创建于 1995 年 4 月，总部设在美国加州的 Mountain View，它是全球软件行业第一家具有商业性质的证书授权机构，提供有顶尖的数字验证产品和认证服务，也是 Microsoft 等许多世界著名软件公司首选的数字凭证(数字标识、数字证书)提供商。该公司网址为 <http://www.verisign.com/>，主页如图 12.4 所示。



图 12.4 VeriSign 公司的主页

VeriSign 提供的认证服务主要包括: Internet 信息服务器(IIS)和 SSL 的 Internet Explorer 浏览器的客户(Client)认证(也称“证书”)、Outlook Express 电子函件系统的 S/MIME 认证以及允许银行进行 128 位加密的 SGC 认证,同时该公司还推出了针对 WAP 服务和 WAP 网关的认证服务。其认证服务的网址是 <http://digitalid.verisign.com/>。

1999 年初,我国开通了首家安全电子函件认证站点(<http://secumail.263.net>)。该站点是由 263 首都在线(<http://www.263.net>)和上海格尔软件有限公司合作推出的,能为电子函件的安全传输提供切实可靠的保证;同时,该站点的建成,结束了中国网络用户只能去国外站点申请网络身份证的历史。该站点采用当今世界上先进的加密算法,使任何一个获得该站点颁发的数字证书的用户都可以轻松地对自己使用的 E-mail 信箱进行加密处理,进而保护自己的通信安全。其网络安全认证系统,还通过了国家公安部的信息安全监测,保证了站点的合法性与权威性。

下面我们列出其他一些可以提供数字凭证认证服务的主要机构,供用户参考。这些认证中心均能提供 VeriSign 的认证服务。

其一, BankGate CA, 其网址是 <http://www.bankgate.com/>; 其二, BelSign NV-SA, 网址是 <http://www.belsign.be/>; 其三, GTE CyberTrust Solutions, Incorporated, 网址是 <http://www.cybertrust.gte.com/>; 其四, KeyWitness Canada, 网址是 <http://www.keywitness.ca/>; 其五, Thawte Consulting, 网址是 <http://www.thawte.com/>; 其六, Entrust 公司, 网址是



<http://www.entrust.com/>，等等。

WAP 数字凭证的认证

前面提到，针对 WAP 安全的数字凭证由于需要在无线网络中传输，需要在具有有限资源的无线用户端进行处理，所以要求使用微型的数字凭证。目前，VeriSign 公司为此专门提供了一种 WTLS 微型凭证来保证 WAP 服务和网关的安全。这种凭证是 VeriSign 公司和诸如 Motorola、Nokia、Phone.com 等 WAP 服务和网关提供商共同合作的结果，能为 WAP/WTLS 提供一揽子的认证解决方案。

电子商务应用需要有凭证的撤回功能，以便保证当服务器的凭证安全有危险或者失效时，用户能够得到判别信息，知道自己正在和一个不安全或非法的服务器进行事务处理，进而作出相应的对策和处理。VeriSign 公司提供的上述解决方案就能方便地识别出那些数字凭证已经失效的服务器或来路不明的非法服务器，这一点对 WAP 用户来说绝对是一件可喜的事情。

然而，WAP 服务器及网关的数字凭证是基于移动用户设备的应用环境的，就现有的资源和带宽，它不可能像在有线网络中那样，能在本地进行证书失效处理，那么，VeriSign 是怎样实现这点的呢？其实很简单，它为用户提供了一种短期的数字凭证 (Short-life Mini-Certificate) 服务，来保证凭证的撤消服务。首先，它为网关或者服务器先颁发一个长时间 (比如一年) 的有效数字凭证，然后在此基础上再发放一个新的短时间 (比如 24 小时) 的数字凭证，那么服务器或者网关就可以使用该短时间的数字凭证来与客户建立会话。如果凭证的所有者想撤回服务器或者网关的凭证，则只要简单地停止发放短期凭证就可以了。

除了支持 WTLS 服务器及网关，VeriSign 的短期微型凭证服务也可以用来支持网关所支持的 SSL 功能和数字签名功能。用户也可以使用微型证书来验证签名的有效性，而无需附加的有效期检测。

12.3 WAP 数字凭证的使用与防范

网上数字凭证使用时，一般都要应用密码技术、数字签名、数字时间戳等安全手段，这一过程目前都是参与双方通过第三方认证中心实现的。而数字凭证在申请、颁发、使用和认证等操作时因为要通过 Internet 与认证中心联系，所以其中又涉及到安全与防范问题。本节就讨论 WAP 客户端、服务器及网关数字凭证的使用方法及安全防范，具体的操作实例

我们在下一节专门给出。

12.3.1 WAP 客户端凭证的使用

WAP 客户端数字凭证的使用主要涉及申请、颁发、获得和签发等几个方面。介绍这些内容之前，我们先来了解一下客户端凭证的实现方法。

凭证实现方法

WAP 客户端数字凭证包含一个专用密钥和数字签名逻辑，客户端的用户使用这种凭证不仅可以给对方提供认证功能，同时还可以实现客户端的数字签名功能。WAP 客户端数字凭证可以应用于大多数的无线设备或手持设备，甚至是内嵌 SIM 3 的智能卡。

我们知道，普通 Internet 的个人数字凭证成功申请后都是安装在客户端的浏览器里。然而，手机等无线设备的客户端浏览器比较特殊，它没有足够的存储能力来存放 WAP 数字凭证，为此，VeriSign 等认证机构通过将凭证小型化和微型化，使得客户端无须存储或者处理相应的数字凭证和公共密钥验证。具体实现的方法是将客户端的数字凭证保存在认证构架中的一个目录或者其他形式的存储设备中。这样，当用户数字签名等需要被验证的时候，就可以从那个地方提取用户的凭证。而且，在这种方式下，我们无须考虑数字凭证规模的大小，所以即便是标准的 X.509 格式的凭证，也可以在无线网络中应用。这种方式最主要的优点是，服务器能在同一个地方完成无线或者有线用户的签名校验。

当然，如果条件允许，数字凭证也可以保存在客户端的用户设备或者 SIM 卡中，而且这样能提供更好的认证速度。

在电子商务中，除了无线用户需要验证自己所连接的无线网关(如无线服务提供商)或者应用服务器(如银行或者股票代理)外，无线用户的对方，如无线服务提供商或银行等也同样需要对用户身份进行认证，由于这一过程通常需要经过 WAP 网关，所以他们可能需要对无线用户及网关同时进行认证。为此，WAP 提供了“用户-网关”认证，该认证建立在 WTLS 基础之上，并可通过 VeriSign 等认证机构申请到相应的数字凭证。这种凭证我们也把它归属于客户端的数字凭证。

凭证的申请

WAP 客户端凭证的申请一般都在普通的 Internet 浏览器上进行。客户端凭证主要分为



两个级别。第 1 级(Class 1)数字凭证, 仅仅能为用户提供个人 E-mail 地址等简单项目的认证, 不能对用户个人的真实姓名等信息进行认证。当用户获得该级数字凭证后, 认证中心即将用户的 E-mail 地址等简单项目的证书列在公共目录中。网上活动中, 凡因 E-mail 地址等简单凭证的丢失、误用或舞弊而引起的经济损失, 认证中心的服务部门将会给予一定数量的经济赔偿。例如 VeriSign 公司就规定这种情况下可给予用户 1000 美金以内的赔偿。

第 2 级(Class 2)数字凭证可以对用户的个人姓名、身份等重要信息进行认证。当用户获得该级数字凭证后, 认证中心也会自动将认证信息列入公共目录中, 并对数字凭证因丢失、误用或舞弊而引起的经济损失进行担保。如 VeriSign 公司对此可给予用户 25000 美金以内的损失赔偿。

凭证的颁发与获得

用户申请客户端凭证后, 认证中心将对申请者的 E-mail 地址、个人身份及信用卡号等信息进行核实, 通常需要几个小时甚或 3~5 天的时间, 核实后即可颁发数字凭证。

数字凭证颁发时是由认证中心给用户发回一个确认的 E-mail, 通知用户有关凭证中的信息, 同时根据凭证的性质及客户端环境, 将该凭证安置在认证构架的目录或存储设备中, 或者安置在客户端的浏览器、SIM 卡中。这样, 用户就获得了自己的数字凭证。

凭证的签发及使用

用户获得客户端凭证后, 就可以通过无线设备或 WAP 服务提供商来选择使用凭证的认证功能和数字签名等功能了。具体设置与选择方法因 WAP 设备、服务器、凭证发布机构的不同而不同, 用户需要参考他们提供的相关资料来确定。

12.3.2 WAP 服务器/网关凭证的使用

WAP 服务器的数字凭证与网关的数字凭证类似, 我们这里就以服务器凭证为例, 讲解凭证的申请、颁发、签发及认证等操作的一般方法。

凭证的可信度

与客户端凭证不同, WAP 服务器数字凭证可以帮助 WAP 服务提供商或应用服务端建立一个网上虚拟环境中的信任度。我们知道, 现实生活中, 一个大商店的整洁环境、服饰一致的雇员和交易中的规范手续, 如收款后出具发票或收据等, 都可以给顾客一种可信度。而在网上虚拟环境中, 服务方与客户方无法面对面地接触, 他们所能凭藉的互相信任的手

段就是企业站点的 WAP 服务器数字凭证。因此，WAP 服务器凭证的建立是比较复杂的。

WAP 服务器凭证包含了服务器的公共密钥并拥有唯一的专用密钥，能授权给确认的站点，以供移动设备内的微型浏览器访问和认证。当微型浏览器的用户想发送一个机密信息给 WAP 服务器时，微型浏览器就会操作 WAP 服务器的数字凭证，一是将接受信息的权限授权给对方 WAP 服务器，即对服务器进行认证，二是使用 WTLS 协议和服务器的公共密钥进行信息的加密。

由于 WAP 服务器拥有自己的专用密钥，所以只有这台经过认证的服务器能够解密用户发来的信息。从而实现信息传输的安全、可靠和对服务器的信任。

一般来说，WAP 服务器凭证的可信度是建立在如下 4 个方面的基础上：

(1) 管理和操作该服务器的企业的可信度。因此，申请凭证时认证中心需要对该企业进行必要的信用调查，这也就等于替企业的客户们进行信用调查。

(2) 验证操作严密而且规范。这通常需要一个权威的认证中心对 WAP 服务器数字凭证的发放进行详细而严格的验证，并对凭证的签收和撤消程序进行严格管理及控制。

(3) 数字凭证操作的技术支持。技术支持越强，WAP 服务器凭证的信用就越高。

(4) 企业及认证中心的设备可靠性。目前比较可靠的安全系统一般包括有多层逻辑访问控制、红外线监视器、生物统计扫描仪以及最新的防火墙等先进的设备和技术等。

凭证的申请验证

申请 WAP 服务器凭证时的验证要比用户个人身份的验证复杂得多。申请时，认证中心将给出一份申请表让企业如实填写，企业可以在认证中心的网页中填写，填完使用电子函件直接传给认证中心；也可以将填写结果打印出来，通过传真或普通邮件发送到认证中心。

该申请表就是认证中心对企业的调查文件，主要包括以下内容：

- (1) 有关企业情况的详细证明项目；
- (2) 有关企业合作伙伴的情况调查；
- (3) 企业营业执照调查；
- (4) 企业 WAP 服务器/网关的技术性能和技术环境；
- (5) 企业 WAP 服务的内容、市场影响力；
- (6) 企业资质调查；
- (7) 企业纳税证明；等等。



建立 WAP 认证服务器

企业通过认证中心的申请调查后，认证中心就会向企业颁发数字凭证，企业在自己的网络站点服务器上安装该数字凭证后，即可建立企业的 WAP 认证服务器。为保证网络认证畅通，企业服务器必须能够支持 SSL/WTLS 的技术要求。建立企业 WAP 认证服务器的过程包括以下几项主要内容：

- (1) 认证中心通知企业已通过认证申请；
- (2) 企业为 WAP 服务器生成一个密钥对；
- (3) 企业向认证中心提出安装 WAP 服务器数字凭证的请求；
- (4) 认证中心通过网络为企业安装 WAP 服务器数字凭证；
- (5) 企业配置和激活 WAP 服务器的安全机制，即可从事 WAP 服务。

WAP 服务器凭证的使用方法

WAP 服务器凭证生效后，企业就能够以被验证的身份在网上与外界通信和进行 WAP 服务、电子商务等活动。服务器数字凭证依据与之绑定的密钥对来表示自己的确定性，用该密钥对进行加密，即可保证该 WAP 服务器的真实身份。

这样，当认证的客户端与认证的 WAP 服务器进行通信或进行网上交互时，客户端就会自动验证企业端的 WAP 服务器数字凭证，而服务器绑定的密钥对又用来加密一个会话密钥 (Session Key)，该会话密钥是用来对服务器和客户端无线设备的会话进行加密的。会话密钥是不同的，每个服务器和客户端的会话都要使用不同的会话密钥。每个会话密钥通常只有 24 小时(甚至更短)有效期，所以，要在认证的 WAP 服务器和客户端通信或交互时进行窃听是比较困难的，这就保证了双方通信或交互的安全性与保密性。

图 12.5 表示了 WAP 服务器与 WAP 客户端通信时互相验证的具体过程。

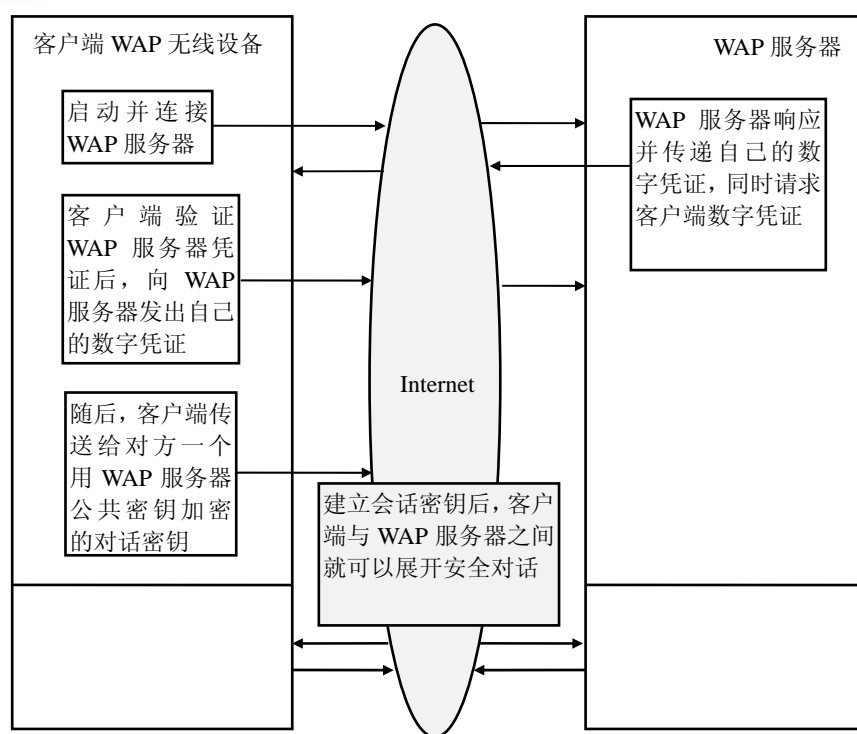


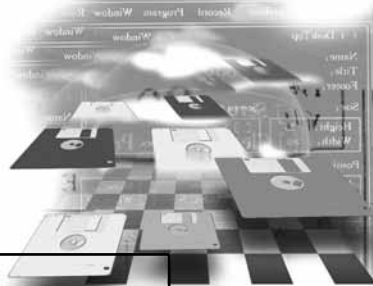
图 12.5 WAP 服务器与 WAP 客户端通信时的互相验证过程

12.3.3 数字凭证认证操作的安全防范及问题

经验表明, 设计和实现一个安全的加密系统是非常困难的。即使是一个最简单的加密算法, 其协议的设计和实现也是一个非常复杂、细致的过程。一个非常小的变化通常会导致一个极其严重的安全漏洞。基于数字凭证的安全电子认证系统虽然已经实用化, 但并不是说这样的安全系统中就不存在可以攻击的漏洞了。

从设计、分析和完成一个加密系统的过程中, 我们可以知道, 如果把密钥解密的过程, 也就是密钥恢复过程加入到加密系统中, 那么就很难保证这样的系统会能像我们期望的那样有效和可靠。因为任何一个潜在的漏洞都有可能让非法用户获取相应的数据并危害我们的正常交易。

例如, 美国的国家安全局可以说是世界上最先进的密码学研究结构, 国家信任它来开发密码系统, 以保护国家最重要的军事秘密和联邦秘密。但其“Clipper Chip”加密系统, 自从 1993 年出现之后, 已经发现了好几个缺点。这并不是说系统设计者无能, 而是因为高度的网络安全性是一个极其艰深的技术问题。一个安全的网络基础设施通常涉及到成千上



万的企业、公司、代理机构，涉及到许多规章、制度，而且还要求世界范围内的法律执行部门以前所未有的程度进行热情合作。

此外，随着应用数字凭证的个人、企业、机构、软件开发者的增多，使用的密钥对会越来越多。2000 年，包括无线网络在内的 Internet 上大约在使用 10 亿多对公用及专用密钥对以及 100 多亿个会话密钥。这些数据将会随着信息技术的发展和网络技术的应用而越来越大，这无疑增加了非法用户解密成功的可能性，降低了网络系统的安全性及可靠性。

网上认证的安全防范还涉及另一方面的问题，即认证人员被贿赂的防范。例如，现有甲、乙两人，甲是认证中心的工作人员，乙贿赂甲，并要求甲以丙的名义签发给他一份数字凭证；这样，乙就可以用丙的名义签署交易文件或催款文件，而任何接收者都会相信这份文件的可信性，因为该文件附有完整的可以验证的数字凭证。为消除这类攻击，可以采用这样一个办法：只有两个甚至更多的认证中心的工作人员合作才能签发一份数字凭证，这样除非攻击者能够买通每一名认证人员，否则无法获取他人的数字凭证。还有一种攻击方法，即买通认证中心的工作人员来伪造数字凭证或伪造旧文件进行网上的非法活动，等等。

总之，为进行安全电子交易和网上信息服务，目前无论是有线网络还是无线网络都已提供了网上安全的保障体系，它是以数字凭证为核心的身份验证、互相出具凭证以及加密报文传送为基本手段；但在实际情况中仍可能会出现各种非法的侵犯行为，因此，建立有效的法律体系，并不断分析可能的漏洞并进行安全防范，是网络安全的一项重要的长期工作，需要政府、企业、用户和技术界、法律界等各方力量的合作与努力。

12.4 WAP 服务器凭证的申请操作与 WAP 服务器配置

相对 WAP 服务器凭证来说，WAP 客户端凭证的申请操作要简单得多，所以我们以 WAP 服务器凭证为例，讲解一下数字凭证的申请操作，并以 Nokia WAP 服务器为例，说明在服务器中安装数字凭证的具体方法。

12.4.1 WAP 服务器凭证的申请操作

我们这里以 Entrust 公司(<http://www.entrust.com/>)的 WAP 服务器认证服务为例，来讲解 WAP 服务器凭证的申请操作过程。整个过程主要需要经过验证企业权限、提交域名和签名

证书、选择联系人、签署文件和确认付费信息等 6 大步骤。

验证企业权限

由于拥有 WAP 服务器的企业的名字将出现在企业收到的数字凭证中，所以 Entrust 公司在发放凭证之前必须保证申请者有权利使用这个名字。这是为了防止未经授权的非法申请者在他们的 WAP 服务器中使用合法企业的名字。

证明申请者有权利在凭证中使用本企业名字的最简单的方法是提交一个 Dun & Bradstreet(D-U-N-S)数字。在美国本土及其他许多国家和地区，大多数企业都有自己的 D-U-N-S 数字。没有该数字的企业可以到 <http://www.dnb.com/> 免费索取一个 D-U-N-S。当然，索取时需要提交有关企业的名称、性质等资料，重要的是要提交企业身份的合法证明。Dun & Bradstreet 确认企业资料正确后，就会免费传给企业一个 D-U-N-S 数字。

这样，在 Entrust 申请 WAP 服务器凭证时，企业要首先提交自己的 D-U-N-S 数字和企业类型信息。Entrust 收到后就会连接到 Dun & Bradstreet 站点，来确认申请者所填的内容是否属实。

然后，企业需使用传真将自己的权利认证传给 Entrust。权利认证文件的要求因申请者所在组织的性质不同而有异，表 12.1 给出了常见类型申请者需要提交的权利认证文件类型。

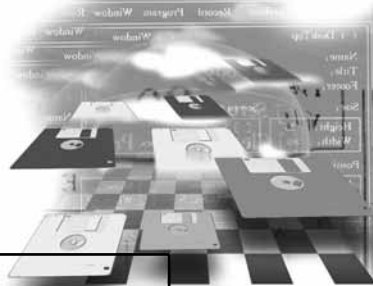
表12.1 常见类型申请者需要提交的权利认证文件

申请者类型	需要提交的文件
公司、企业、股东、所有者	公司的注册文件；或者是经过有关方面盖章的股东证明或所有权的复印件
政府部门或机构	一封由部门总负责人签字的原始信件。信件中必须包括部门的相关信息和当时负责人所在的职位
非政府组织	一封由主席、首席执行官或者总管理人签署的申请信
大学	一封由所在院校的院长或者副院长签名的申请信。信中必须包含所在院校的信息
商业组织	一份有关商业组织的注册文件，税务和纳税人证明，或者是官方相应的书面证明，表明申请者有权利使用该组织的名称
其他类型申请者	需要和 Entrust 联系商讨应该提交的具体文件

提交域名和 SSCR 等信息

接下来，申请者还要向 Entrust 提供自己的域名和自签名请求证书 SSCR(Self-Signed Certificate Request)。同时需要选择证书的有效期限、口令，并指定 WAP 服务器使用的软件系统。

(1) 域名。之所以要提交域名信息，是因为企业收到的数字凭证中包括有自己 WAP 服



务器的域名。提交域名信息还要同时提交域名所有者名称、企业名称及其所在地址。

(2) SSCR。自签名请求 SSCR 包含有企业 WAP 服务器的公开密钥, 以及关于服务器识别名 DN(Distinguished Name)的信息。SSCR 是由企业生成的, 它是针对企业 WAP 服务器的签名证书。生成 SSCR 的操作并不复杂。如果企业的 WAP 服务器是租用 ISP 的服务器, 那么 ISP 会在线地提供给企业一个 SSCR。如果 WAP 服务器是自己的, 那么可让 Entrust 来生成 SSCR。不过 Entrust 并不能支持所有类型的 WAP 服务器生成 SSCR, 它支持的服务器类型清单可参见 <http://www.entrust.net/customer/wapserver-support/index.htm>。好在绝大多数企业所用的 WAP 服务器都不会超出这个清单, 所以企业不必为生成 SSCR 而发愁。企业使用的 WAP 服务器通常都提供有如何生成 SSCR 的指导信息, 操作时可以参考这些信息。

一般来说, 生成 SSCR 时企业需要在线提交相应的资料, 主要涉及企业整个组织的情况和企业 WAP 服务器的技术资料。这些信息将用来产生 WAP 服务器的识别名 DN。具体的输入项目有:

- 国家代码。这是两个字符的 ISO 标准缩写, 代表了企业所在的国家。例如, US 代表美国, CN 代表中国。
- 地区或者省份。即企业组织总部所在的地区或省份。需要输入详细名称, 不能使用缩写。
- 位置。指企业总部所在的城市名。
- 组织名称。指企业的注册名称。该名称必须和 WAP 服务器拥有同样的域名, 不能简写, 名称中也不要使用 <、>、~、!、@、#、\$、%、^、*、/、\、(、)、? 和句点(.)等特殊字符。
- 单位。即将要使用数字凭证的部门或小组的名称。
- 域名。指企业的 WAP 服务器将要使用的 URL 地址, 不能使用 IP 地址。

在最后上交表单时, 上述输入的内容将粘贴到表单中, 并同表单一起传给 Entrust, 这些数据将来都会打包到 Entrust 发放给 WAP 服务器的数字凭证中。

生成 SSCR 时会同时生成一对密钥, 其中公共密钥将放入到 SSCR 中并且提交给 Entrust 认证中心进行认证, 专用密钥将保存在企业的服务器里。企业一定要注意将专用密钥保存好, 否则如有丢失或者损坏, 都会导致 WAP 服务器无法使用申请到的数字凭证。

下面就是一个 SSCR 的文本内容:

-----BEGIN CERTIFICATE REQUEST-----

AAAACkZvckVudHJ1c3QIAIC4GoFpe7WH4OFMSGUL

IA6s4q1M/i61tUX3w38Ndl5LN72uVjXisXXCI5WT

aY3scejJK4Q6LByFSaY7XDDmkAPwr6gGKNAe56SL

ULXeA9R81DDFB+siSb/cEP74LrU4tkMBHPnuaFn5

q9XLof/ptAttACHJQ9HhrvverzQ2E8nrDQADAQAB

-----END CERTIFICATE REQUEST-----



注意，生成 SSCR 时提交的组织名称应当和前面权利认证文件中提交的企业所用组织名称保持一致。如果不一致，就会增加 Entrust 的认证环节，从而拖延企业的申请时间。

(3) 凭证的有效期。生成 SSCR 后，企业还要选择 WAP 服务器数字凭证的有效期。Entrust 提供了一年或两年的有效期。这两种方式都提供有很好的安全保障，但是大多数企业都是选择两年期的有效期，目的是减少与 WAP 相关的认证工作。我们的建议也是选择两年期的认证。

(4) 凭证口令。申请凭证时企业还要指定一个口令密码。如果将来企业想撤销凭证，那么当和 Entrust 支持部门联系的时候，企业必须输入口令才可以完成需要的操作。为了安全起见，建议使用至少 8 个字符，并且至少有一个小写字母、一个大写字母和一个非字母表的字符(如%、!等符号)来作为口令。好的口令应当是易记而难猜。口令设置好后，最好写下来，并将它保存在安全的地方。

(5) 输入 WAP 服务器类型和所用软件系统环境。具体可以根据 Entrust 给出的列表清单填写。

选择联系人

申请 WAP 服务器数字凭证时，企业还需要提交供 Entrust 与企业联系的人员名单及相关资料。主要有 4 种联系人。

(1) 授权人。通常是企业的一位高级成员，具备申请服务器数字凭证的权利。凭证申请到后，该授权人将会收到凭证的副本。如果处理企业申请时遇到一些更深层次的问题，Entrust 将会直接与该授权人取得联系。

授权人的资料主要需要提交姓名、所在企业名、电话号码、职位名称及 E-mail 地址。

(2) 技术联系人。此人主要负责数字凭证的更新和升级等工作，应当是每天负责安装了

凭证的 WAP 服务器运行的那位技术人员。如果企业的服务器是由第三方的 ISP 负责管理，那么应让第三方安排人作为本企业的技术联系人。

技术联系人的资料主要需要提交姓名、所在企业名、电话号码、职位名称及 E-mail 地址。

(3) 安全联系人。即安全负责人，需要熟悉安全方面的事务，通常与技术联系人是同一个人。Entrust 将向安全联系人发送有关安全的信息和 WAP 安全警告。

安全联系人的资料也主要需要提交其姓名、所在企业名、电话号码、职位名称及 E-mail 地址。

(4) 帐单联系人。即负责向 Entrust 付帐的人。显然这是必不可少的。

帐单联系人的资料主要需要提交其姓名、地址、城市、省份或地区、邮政编码、国家、电话号码及 E-mail 地址。

确认

完成上述内容后，Entrust 将会返回给申请者一个确认页，其中包含有申请者每一步所输入的信息，如：D-U-N-S 数字、企业类型、域名、识别名、有效期、WAP 服务器类型和软件系统，以及授权人、技术联系人、安全联系人、帐单联系人的相关资料等。这个页将给申请者在提交前再次验证输入内容的机会。如果想修改某些内容，可单击页面中的“编辑(Edit)”按钮来修改。

签署文件

完成确认页的操作后，申请者将被要求阅读 Entrust 针对 WAP 服务器凭证的认证服务协议条款。申请者必须完成同意这些条款才能进行下一步操作，才能申请到服务器凭证。

付费信息

最后需要完成付费信息。企业可以通过在线付费，比如使用 American Express、Visa 和 Master Card 等信用卡，也可以通过银行付费。为此申请者要向 Entrust 提交付费的信用卡号或银行账号。当 Entrust 发放企业的 WAP 服务器凭证后，就会收取企业的费用，并给企业开具收据。

申请之后

完成申请之后，申请者将会收到 Entrust 给出的序列号，请记录这个号码。申请者可以使用这个序列号来在线跟踪申请的进展状况。而且，技术联系人和授权人也将收到一个

E-mail, 告知他们, 申请正在被考虑之中。如果申请成功, Entrust 公司将发来一个 E-mail, 其中含有一个 URL 地址, 通过该 URL 即可下载并安装申请到的 WAP 服务器数字凭证。随后企业可以使用该凭证开展 WAP 服务或业务了。

12.4.2 Nokia WAP 服务器中的凭证安装

针对不同的 WAP 服务器, 数字凭证的安装方法一般来说是不一样的。我们这里以 Nokia 提供的 WAP 服务器系统为例, 说明安装数字凭证的具体方法。

我们假设企业所用的 WAP 服务器安装的是 Nokia WAP Server v1.0 系统, 拟安装 Entrust.net WAP Root 数字凭证。该数字凭证的申请可采用前面介绍的方法操作。

配置 Nokia WAP server 的操作主要有 3 个步骤: 其一, 创建一个流量映射, 让内建的文件服务组件(FileServlet)来处理本地文件; 其二, 配置 FileServlet 让它来处理 Entrust.net WAP Root 证书; 其三, 建立一个 WML 服务页面连接来操纵 WAP Root 凭证。

建立流量映射

操作方法是, 在任务栏上单击“开始”按钮, 从中打开“程序”菜单, 并选择“Nokia WAP Server”子菜单中的“Nokia WAP Server Manager”命令, 屏幕上随后就会出现 Nokia WAP Server Manager 管理窗口。

从其菜单条中选择打开“Servlets”菜单, 并选中“映射(Mappings)”命令。Servlet Mappings 的控制面板就会出现在屏幕上。利用该面板, 即可为 WAP 服务器建立一个新的流量映射。缺省的映射是所有的“http://”请求都由 HTTP Manger 来处理, HTTP Manger 负责向 Internet 发送请求。对于本地安装的 Entrust.net WAP Root 凭证, 我们必须创建一个到 FileServlet 的新的文件请求。

为此可单击其中的“创建(Create)”按钮, 屏幕上就会出现一个新的 Mapping 对话框。在其中的 URL 一输入框中输入“http://myhost/”。这里的 myhost 是指当前的主机名。然后从“Servlet”列表中选择“FileServlet”, 最后单击“确定(OK)”按钮。这样就建立了一个新的映射。以后任何到 WAP 服务器的“http://myhost/”请求都将被引导到 FileServlet。

如果映射创建之后需要修改, 可先从 Servlet Mapping 窗口选择需要修改的项目, 然后单击“编辑(Edit)”按钮, 在随后出现的 New Mapping 对话框中即可编辑该映射。

需要指出的是, 在流量映射正确创立的情况下, 我们必须在 FileServlet 中指定 WAP 服



务器数字凭证的文件类型(MIME)并正确配置,使得 WAP 服务器可以正确地将文件发送给手机等移动设备终端。MIME 文件类型其实就说明文件中有 WAP Root 凭证的内容。

配置 FileServlet

保留 Nokia WAP Server Manager 窗口打开,并关闭其他所有窗口和对话框。然后从 Nokia WAP Server Manager 的菜单条中选择打开“Servlets”菜单,并从中选择 FileServlet 命令,这样可以打开 FileServlet 对话框。从中单击“添加(Add)”按钮,屏幕上即出现“添加属性(Add Property)”对话框。

在该对话框,从其“Name”文本框中输入“.cer”,从“Value”文本框中输入“application/vnd.wap.wtls-ca-certificate”,然后单击“确定(OK)”按钮关闭当前对话框,完成将 WAP 凭证的文件类型加入到 FileServlet 的操作。

这时返回到 FileServlet 对话框,单击其中的“确定(OK)”按钮即可关闭该对话框,完成 FileServlet 的配置操作。

建立操纵 WAP Root 凭证的 WML 服务页面

最后,我们还需要把 Entrust.net WAP Root 凭证文件加入到 WML 页面中,以使得终端用户可以选择操作 Entrust.net WAP Root 凭证文件。

下面的 WML 卡片程序实现了对 Entrust.net WAP Root 凭证的连接。其中我们假设 Entrust WAP Root 凭证的名称是 EntrustRoot.cer,该凭证保存在当前 WAP 服务器的根目录中。程序代码如下:

```
<card id="CaRoot" title="Entrust.net Root">
  <do type="prev" label="back">
    <prev/>
  </do>

  <p>Import Entrust.net root certificate:<br/>
    <small>(requires Nokia 7110/7190)</small><br/>
    <small>-<a href="http://myhost/EntrustRoot.cer">CA Root</a></small><br/>
  </p>

  <p align="right">
    <small>? 2000 Entrust Technologies</small>
  </p>
```



</card>

这样，Nokia WAP 服务器就配置好数字凭证，终端用户就可以下载和使用 Entrust.net WAP Root 凭证了。

本章小结

本章我们讨论了 WAP 的安全问题和实现方法。由于网络安全问题本身就比较复杂，所以考虑到大家的快速切入，我们首先讲解了数据加密原理与实现方法，介绍了基于单钥技术的传统加密方法、改进的传统加密方法和基于双钥技术的现代加密方法。在此基础上，我们介绍了实现 WAP 安全的一般方法，如数字签名、数字时间戳、数字凭证、WAP 微型凭证、认证中心等。然后，我们更具体地讲解了 WAP 客户端凭证、WAP 服务器/网关凭证的使用方法与防范问题，并以 WAP 服务器凭证为例，介绍了数字凭证的申请操作和 WAP 服务器的配置方法。

除了需要重点掌握 WAP 凭证类型及其申请、使用和配置方法外，大家还要注意理解基本的加密原理，了解双钥技术的特点，熟悉数字凭证、数字签名、认证中心等概念和它们的运作方式。

参 考 文 献

- [1] Ericsson Corporation. *Mobile Phone R380 Design Guidelines for WAP Services*, <http://www.ericsson.com/WAP/developer/>, 1999.
- [2] Ericsson Corporation. *Mobile Phone R320 Design Guidelines for WAP Services*, <http://www.ericsson.com/WAP/developer/>, 1999.
- [3] Ericsson Corporation. *Mobile Phone MC218 Design Guidelines for WAP Services*, <http://www.ericsson.com/WAP/developer/>, 1999.
- [4] Nokia Corporation. *WML Reference*, Version 1.1, <http://www.forum.nokia.com/>, 1999.
- [5] Nokia Corporation. *WMLScript Reference*, Version 1.1, <http://www.forum.nokia.com/>, 1999.
- [6] Nokia Corporation. *Developer's Guide, NOKIA WAP TOOLKIT*, Version 1.2, <http://www.forum.nokia.com/>, 1999.
- [7] Phone.com, Inc. *WML Language Reference*, UP.SDK Version R4.B2, <http://www.phone.com/>, 1999.
- [8] Phone.com, Inc. *UP.SDK Developer's Guide*, UP.SDK Version R4.B2, <http://www.phone.com/>, 1999.
- [9] Phone.com, Inc. *UP.SDK Tools and APIs Reference*, UP.SDK Version R4.B2, <http://www.phone.com/>, 1999.
- [10] Phone.com, Inc. *WMLScript Reference*, UP.SDK Version R4.B2, <http://www.phone.com/>, 1999.
- [11] Phone.com, Inc. *WMLScript Developer's Guide*, UP.SDK Version R4.B2, <http://www.phone.com/>, 1999.
- [12] WAP Forum. *Wireless Markup Language Specification*, <http://www.wapforum.org/>, 1999.
- [13] WAP Forum. *WMLScript Specification*, <http://www.wapforum.org/>, 1999.
- [14] WAP Forum. *WMLScript Standard Libraries Specification*, <http://www.wapforum.org/>, 1999.
- [15] WAP Forum. *Wireless Application Environment Specification*, <http://www.wapforum.org/>, 1999.
- [16] WAP Forum. *Wireless Application Protocol Architecture Specification*, <http://www.wapforum.org/>, 1999.
- [17] WAP Forum. *Wireless Session Protocol Specification*, <http://www.wapforum.org/>, 1999.
- [18] W3C Proposed Recommendation. *Extensible Markup Language (XML)*, <http://www.w3.org/TR/REC-xml/>, 1999.
- [19] 曹建,《Internet 与 E-mail 安全防范实用技术》,成都:电子科技大学出版社,1999年.
- [20] 曹建,《电子商务与网上经营》,成都:电子科技大学出版社,2000年.