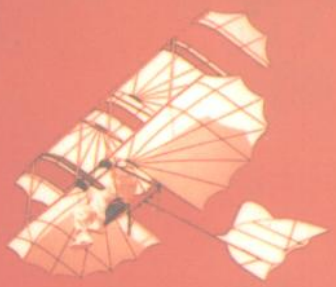
王佑中 著



Web 动态技术入门

INTERNET实务系列丛书

机械工业出版社

Internet 实务系列丛书

WEB 动态技术入门

-CGI，Java & Java Script 探索

王佑中 著

机械工业出版社

内 容 简 介

本书较详细地介绍了应用网络动态技术领域中最热门的工具 CGI 和 Java & Java Scipt。本书以生动的语言与读者一起探讨了 CGI 程序技巧，Java & Java Script 在网络环境下做为一种对象式语言的应用以及相互间的差别。着重讲述了 Java 的执行环境、图形对象与 Java、Java 的应用程序、HTML 的内部巨类——Java Script、Java Script 程序库、以及 DEAL 与 TCL 的快速入门等。

本书适合即将或已经进入 WWW（全球信息网络）的用户，以及对此感兴趣的计算机网络爱好者阅读。

本书版权登记号：图字： 01-96-1266

图书在版编目（CIP）数据

WEB 动态技术入门：CGI，Java & Java Script 探索/王佑中著。—北京：机械工业出版社，1997. 1

（Internet 实务系列丛书）

ISBN 7-111-05456-3

Ⅰ. W… Ⅱ. 王… Ⅲ. 全球网络：互连网络，Internet-检索系统-程序语言 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（96）第 20290 号

出 版 人：马九荣（北京市百万庄南街 1 号　邮政编码　100037）

责任编辑：王　斌

三河永和印刷有限公司印刷 · 新华书店北京发行所发行

1997 年 1 月第 1 版　1997 年 4 月第 2 次印刷

787mm×1092mm　1/16 · 13.25 印张 · 330 千字

6001-9000 册

定价：23.00 元

前　言

过去一年来电脑界最受瞩目的技术恐怕非 Web 莫属了，从最开始的 Mosaic 开始带起这波风潮，每个人开始做自己的 Homepage，这使得市面上有关 HTML 写作的书约在 10 本以上。随着 Web 由网络上玩家的手中慢慢地进入商场之中，对于品质的要求越来越高，过去静态的画面已不能满足大家的需求。各种有关动态文本的技术也一一出现，最早是在 CGI 上面做文章，但很快大家也了解到 CGI 其实并不适合做这一类的工作，因为它需要太多的网络及服务器的资源。

Java 和 Javascript 在此时出现在大家的眼前，它们立刻被接受为最适合这一类工作的技术。Java 的出现使得 Web 的地位逐渐的被提高，有些看法认为它很适合做一个跨平台操作系统的基础，也许我们可以使用一个浏览器（Browser）来代替传统的操作系统。

当然这一切都还在未定之中，甚至连 Java 本身都还不是一项很成熟的技术。但这开启了一个充满不确定性的未来，Netscape 和 Yahoo 一夜成名的例子使得 Web 的发展更加上了一些刺激性。

也许这一波技术风潮的高峰已经过去，接下来的是一连串平稳的技术发展。天才的创意可能已经不是那么容易出现，需要的也许是对技术本身更进一步的了解。本书的目的是为在这一波风潮中出现的一些技术做一些整理，除了这些技术本身使用上的说明外，对技术背后的想法及细节我也试图加以说明。当然这些技术都实在太新了，新到你我都一样的陌生，我所能做的只是试图尽力收集各种资料而已。

这本书也许缺少了一般书会有的"真实范例"，因为我认为那种范例除了占去书籍大量的篇幅外并不能增加读者多少的了解。反而一些小巧的范例更能让大家比较容易将重点集中在焦点之上，所以本书中的范例都十分小巧，甚至是有点将问题过分简化。

这本书大约可分成三个部分，第一个部分是针对 CGI 及相关技术如 HTTP 协议、MIME 协议、CGI 协议等的说明。并且将重点放在如何使用 perl 来完成这些工作，在这里我假设大家都会使用 perl，在本书的最后将会有一整章的篇幅介绍 perl 程序的编写，我相信这可能是目前唯一有关 perl 比较完整的中文说明了。

而第二部分把重点放在 Java 及 Javascript 上，由于篇幅的关系，我把整个 Java 程序库的说明准备放到下一本书中再加以说明，在这本书中则尽力说明 Java 这个语言本身有关的事项。其实我认为 Java 做为一个一般性对象语言的潜力并不小于在 Web 上的应用，使用它可以轻易的做出一个跨平台的程序系统。这一部分另一个重点则在于 Javascript 上，这个语言是 Web 族群中最年轻的一员，甚至于它本身还是一个不太成熟的技术，在这里介绍的目的是让急于赶上技术前端的人可能很快的开始使用它，Javascript 绝对是未来 Web 列车中位于关键位置的一员。因为它的功能是用来控制系统中诸多元件的控制语言，在将来诸多技术一一进入 Web 后势必有一种语言出来总管这些对象，这个语言当然就是 Javascript 了。

本书的最后一部分则提供了 UNIX 上常见的两种命令语言 perl 和 TCL 的介绍，perl 是 CGI 程序的重心，自然不能不提。而 TCL 则可以用来和 TK 合作很简易的完成一个窗口程序，在 CGI 中使用它可以在客户端显示出一个 X 窗口用来展示各种画面，虽然这并不是一个很安全的做法，但对大多数的使用者而言大概还可以接受。

目　录

第一篇　了解及使用 HTML 语言

第二篇　CGI 程序技巧

第三篇　Java

第四篇 PERL 与 TCL 快速入门

第一篇

了解及使用 HTML 语言

第 1 章　全球信息网络（World Wide Web）的基础

WWW 是 Tim Berners-Lee（TBL）在差不多五年前所提出的概念，至少在使用层面上获得了很大的成功。但它是什么东西相信很多人还是很模糊的，不过看完这一章，相信你会对它有很不同的看法。

1.1　什么是 WWW

不知道从多久以前开始，我们常可以在电影中看到如下的情节。主角向一个电脑天才问道："你能告诉我有关 XXXX 的事吗？"电脑天才回答："没问题！请等一下"。

这时电脑天才便运指如飞地在键盘上打入一堆谁也看不懂的命令（谁晓得他们是不是乱打的），然后在屏幕上显示一行一行的数据。然后他会向主角说道

"根据所有数据库内的数据，关于 XXXX 只有下面的数据"

不过这时屏幕上突然出现了一些不寻常的信息。天才眼睛一亮，用非常沉稳的口气说道："不过它有一个数据被密码保护住了，这个密码很难解，想必是很重要的数据……"

这个天才所说的所有数据库的数据当然不可能存在银幕上看来破破烂烂的电脑中（又是一个常见的惯例，电脑天才的电脑绝对长得"不好看"），它们是存在全世界各个不同地方的电脑之中，电脑天才是通过网络来查询这些数据库的。当然存取这些数据的方法就是电脑天才的秘密了（否则每个人都会，那他还叫"电脑天才"吗）！

这种理想的起源远远早于这些电影，在传统传说中的万事通就是扮演这种角色。它们的工作是收集数据，任务是回答问题。当然他们的答案可能都不太清楚，甚至不太正确，因为他们扮演的只是传达消息的任务。故事的主人翁当然就要负责由这些密语中找出事情的真相。

因为人的生命有限，没有人能真正知道所有的事情，更没有人能记得所有的事情。即使在古时候简朴的社会中这也是不太可能的事，更何况在现在的社会。但当电脑出现后，人们开始想象一个能知道所有东西的机器。因为它是机器，所以它不会死，因为它是机器，所以它不会忘记。但不要忘记，它是机器，所以它不会思考，会思考的是你。World Wide Web，通常被简称为 WWW，提供了一个人们自古以来想象中的能力，但它只能提供数据，它不会分析所得到的数据是对是错。

WWW 可以让你由一个被称为 HomePage 的文件开始，将世界上的数据互相连接，你不必把所有的数据都放在自己的电脑之中，因为那是不可能的。即使你有那么多的空间存储它们，但数据是活的，你不可能期待某国的元首永远不会改变（即使它们自己多么希望如此），你不可能认为某地的气候每年都一样。你可能不断去更新这些数据吗？当然不可能，美国的总统是谁当然要由他们自己决定（不过美国到是蛮喜欢帮别人选择总统），所以有关它的数据当然应该由他们自己维护。WWW 的精神就是让每个人维护自己的数据，而互相为别人提供自己的数据。如此一来大家便都有全部最新最正确的数据了。

在 WWW 之前其实已经有很多类似的想法了，但为什么 WWW 会如此流行呢？除了近年

来网络环境的改善之外，WWW 提供了图形的能力是一个重要的关键。我们可以在电脑上直接看到宇宙飞船升空的画面，或是哈柏望远镜最新拍到的宇宙奇观。整个动作可能只是用鼠标在屏幕上点几下而已。

随着时间的演进，WWW 的能力也不断地在扩充。最早期的 WWW 实际上只比以往的系统多了一些有限的图形能力，现在的系统已经越来越多姿多彩了。WWW 主要由

- 超文本传输协议　HyperText Transfer Protocol（HTTP）
- 超文本排版语言　HyperText Makeup Language（HTML）
- 通用网关界面　Common Gateway Interface（CGI）

等三个部分组成，它们分别位于客户端（client）和服务器端（server）上。在服务器端的程序被称为服务器程序，而在客户端的程序常被称为浏览器（browser），它们分别扮演数据的提供者和合成使用者看到画面的工作。以下几节就分别说明这些协议及程序的内容。

1.2 超文本传输协议（HTTP）

HTTP 是一个专门为 WWW 的服务器和客户程序间交换数据所设计的协议。根据 TBL 自己的说法："HTTP 是一个无状态的面向对象式协议"。所谓无状态是指不论在服务器端或是客户终端都不必为建立起的连接（connection）存储任何的数据，两端都可以由一次传进来的数据中知道应如何响应。这种协议由于不必存储状态，所以当接到对方的信息后只要做一次响应便可以不再理会它，直到它下一次发出信息为止。如此对服务器而言，可以很容易地为多个客户服务，而不必有 multiplex① 的工作，故这个协议不管在使用或操作上都是很简单且易用的协议。

对于一个多媒体的系统而言，用来传送数据的协议必须有很快的响应时间，也就是说我们不可以让使用者等很久却不告诉他发生了什么事情。虽然不见得在所有的情况下 HTTP 都可以完全达到这个效果，但即使在使用 14.4KModem 来传输数据，依然可以得到尚可接受的效果。

如果不打算自己写一个浏览器或服务器，通常可以不必太在意这个协议。有兴趣的人可以参阅：

http：//www.w3.org/www/Protocols/HTTP1.0/HTTP1.0-ID_1.html

在后面我将会专门说明这个协议。

1.3 超文本排版语言（HTML）

HTML 是一个 SGML 的应用程序，SGML 是一个通用的文字图形编排语言。也许我们应该称它是一个准语言（metalanguage），因为它本身并没有固定的语法，它只是一些定义语法用的规则。各位也许可以把它和 YACC 相比较，不过它与其说是一个语言，不如说是一个定义语言的系统。它可以很精确地定义出一个语言，通过它我们可以轻易地验证我们写出的 HTML 程序是一个没有语法问题的程序，因为通常浏览器是不会考虑太多如何处理错误的问题。

在 WWW 被提出的同时，HTML1.0 也随之被提出。不过当时它只是一个很简单的语言，

① 所谓 multiplex 是指同时等待好几个网络连接时，必须不断地检查每一个连接是否有数据进来。如果系统没有提供同时监控好几个连接的能力，则一定会因为不断地检查连接状态而使系统性能降低。

几乎只提供显示标题的能力和很有限的图形能力。但随着 2.0 的提出，HTML 被 IETF (Internet Engineering Task Force)支持而成为一个 Internet 标准。不过即使是 2.0 版的 HTML 仍然缺少了很多多媒体系统应具有的能力，如图文混合、表格、数式和更精细的文字排版控制等。所以 3.0 版被期待把这些东西一一加入，目前已经有一个 3.0 版的草案出现，不过它目前还在实验的阶段，但可以期望在不久的将来就会进入草案阶段。

除了 HTML3.0 版之外，在 netscape 中也提出了一个不同的版本，它是基于 HTML2.0 的扩展。目标和 3.0 版有很多重叠，事实上在语法上也有很多相同之处。不过使用这些扩展的结果很可能使得兼容性变差就是了。

1.4 Common Gateway Interface (CGI)

有时候把所有的数据原原本本的送给客户程序并不是一件好事，例如一个存放所有论文的 WWW 服务器，如果把每一篇论文的内容，即使只是摘要放在网络，对使用者而言可能等于没有。它们可能希望能通过关键字来找出它们所要的论文。也许我们可以在服务器内提供一个功能来关键字搜寻的工作，但诸如此类的需求非常的多，彼此间的差异也可能很大，把每实现一个功能都做进服务器内并不是一个好主意。所以出现了 CGI 这种界面，它基本上是服务器和一个外部程序间的通讯界面。这个程序可能是一个 C 语言的程序，也可能是一个由 shell script 或 perl 所写成的程序。服务器不会管这个程序是由什么语言写成的，它只是简单把这个程序所产生的 HTML 文件或其他合乎 HTTP 规格的文件传回给使用者。你可能不经过 HTTP 而直接由程序传送给客户 程序，如有一些 WWW site 上提供执行一些程序并把结果显示在使用者的 X 终端机上即是如此。

CGI 的功能是由 HTML 语言中的<FORM>和<ISINDEX>命令所定义的，详细的协议请参考第 3 章。

1.5 WWW 如何工作

对于一般用户而言，WWW 指的是 netscape、Mosaic 或其它的浏览器。它们基本上是 WWW 的客户程序，我们只要给这个程序一个 WWW site 的 HomePage 的 URL，它就可以找到对应的数据并且把它们适当地显示在屏幕上。

URL 可以看成是在 WWW 所连成的网络中的一个路径，让我们来看一下图 1-1 在这个图中的 C 是一个路径名称，它是相对于 WWW site 所设定的 WWW 根目录。一个 WWW site 中所有的 HTML 文件都是放在这个目录之下①。

而 A 则是 WWW site 的 Internet 地址，你可以用数字或是文字名称来指定。如果你没有指定，也就是说省略了 A。假设是使用在 local 端机器上的文件系统，此时的 WWW 根目录和用户眼中的根目录是相等的。

如果 A 被指定，那浏览器会发出一个 HTTP 调用。如果没有特别指定的话，就使用 port 80②。你可以用下面的 URL 使用一个连接在 port 8080 上的 WWW 服务器。

当服务器接到这个 HTTP 调用后，它会把整个 HTML 文件传回给客户程序。要注意服务器是传回一个 HTML 文件给客户程序，至于如何解释这个文件就是客户程序自己的事了。所

① 虽然你可以用'…'来违反这个规则，但有必要如此做吗？
② 在 UNIX 上一个机器可以同时连接到很多的服务器上，每个服务器使用不同的 port 彼此分辨。

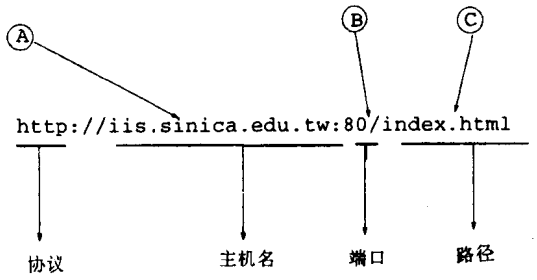


图 1-1 URL 的格式

以一个浏览器可以被看成是一个 HTML 文件解释器，或预览程序（如果把 HTML 当成是一种文字编排语言）。

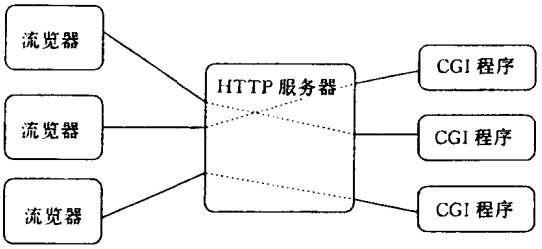


图 1-2 CGI 调用的流程

而 CGI 则是另外一回事了（图 1-2），如果用户在传回的 HTML 文件中的<FORM>或<ISINDEX>上做了选择的动作。浏览器会传送一个调用给服务器，服务器在接到这个调用后会执行相对应的外部程序，并把用户传送过来的数据，如在<FORM>表格中填入的数据设在环境变量中送给外部程序。外部程序在处理完后将结果写在一个由服务器指定的文件中，这个结果通常是一个 HTML 文件，服务器再把这个文件传回给用户做为执行的结果。所以 CGI 可以看成一种动态产生 HTML 文件的机制，我们几乎可以使用它来做出任何的效果。有一些程序如 satan① 就完全使用 HTML+CGI 所组成的界面来做成。这样便把 WWW 当成一个用户界面产生器来使用了。

① 一个用 perl 写成用来检查网络安全的程序。

1.6 Java 和 Javascript

如果说 CGI 让服务器活了起来，那 Java 和 Javascript 就是让客户程序活起来的工具。在它们还未出现之前，浏览器只是一个用来将 HTML 文件转译成给用户观看的工具，就好象在 UNIX 中 more 这个程序就是一个最简单的浏览器，虽然它能处理的只是最简单的 ASCII 协议而已。

但是当它们出现之后，浏览器一夕之间成为一个工作环境、一个图形系统或夸张一点说是新的操作系统。其实从它的功能来看并不为过，因为在 Java 所需的虚拟机器须能管理文件系统和图形显示系统、可以管理程序所需的内存、提供程序执行所需的资料、管理行程及线程（thread）和提供网络的功能。这些正是一般操作系统所需提供的功能，而且事实上操作系统所做的事就是这些而已。

在图形操作系统泛滥的今天，用户界面的程序越来越复杂，而不同平台之间的差异性足以使程序员考虑重写。想要维持一个多平台的程序所需付出的代价极高，所以已经有人在想象使用 Java 做为程序系统，而用 Java 虚拟机器做为操作系统的可能性。因为至目前为止所有的操作系统在设计时都不把用户界面考虑在内，想象一下如果以后 AutoCAD 的程序都用 Java 的形式出现，而使用一个支持 Java 虚拟机器的操作系统（可能是 Unix+X 或是 Windows 甚至是 DOS 加上图形界面）。如此一般的用户可以在 PC 下执行 AutoCAD，而需要更高速度的用户则可以使用 RS6000 的高速工作站来执行它。

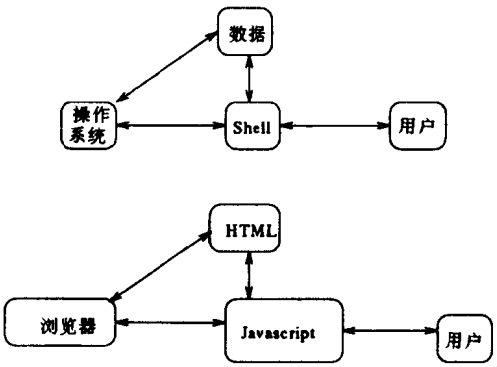


图 1-3　Javascript 在浏览器中的角色

这也许是一个梦想，但如果它成真则可能会彻底改变目前软件界的状态。目前，PC 软件业几乎被微软所垄断，有些厂商，包括 Sun 本身在内都想要打破这种形势。Java 或类似的技术则成为一个焦点，因为如果 Java 成为程序语言的主流，则微软所拥有的 Windows 优势将在一夕之间化为乌有。因为人们关心的不再是程序是否与 Windows 兼容，而是程序是否是由 Java 所写成的。拥有一个 Java 的程序就可以在 PC 或是 SUN 工作站上执行相同的程序，如此用户可以根据预算和需要来选择不同的工作平台和操作系统，无论它们是什么都可以有相同的功能。

以前 SUN 花了很大的工夫想做一个 Windows 的模拟器，主要的原因就是想吸引原来在 Windows 的用户使用 SUN 的机器。但是由于所有的技术都掌握在微软的手中，且微软可以随时更新系统或在程序中使用一些未公开的系统调用让 SUN 疲于奔命。如果 Java 一旦成真，Windows 的用户可以轻易地在需要的时候转换到 SUN 上。但由于 Java 采用公开的策略以吸引更多的用户，这些用户除了 SUN 外也可以转换到其他的平台之上。所以有人就质疑 SUN 发展 Java 的动机，SUN 副总裁给了一个很值得玩味的答案—"SUN 的目的只是在打破目前软件业被垄断的情况"。也许这就是 Java 最好的一个注解吧！

本书的另一个主角是 Javascript，它的功能你可以想成是操作系统和 shell 的关系。Javascript 本身并不提供显示数据的能力，它没有任何函数可以在窗口中画出任何影象。但它提供了控制浏览器行为的能力，你可以使用它在 HTML 文件和用户中间做交谈。

由上面你应该可以了解 Java 和 Javascript 只是在名称上有所类似，它们的功能是很不相同的。我们可以用 Java 写一个很复杂的程序，但不能希望用 Javascript 做相同的事，因为 Javascript 只是一个和 shell script 目的相似的语言而已，没有人会用 shell script 写一个几万行的程序吧！

第 2 章　HTML 语言总览

虽然 HTML 语言并不是本书的重点，但是本书的内容与它息息相关，所以我在本章中以简短的叙述介绍一下这个语言。

2.1　HTML 设计概念

HTML 语言由 1.0 简单的规格开始，至今它已经不再是一个很简单的系统了。它试图支持各种不同的需要，包括表格、数式及其他更复杂的排版策略都一一地被加了进来。3.0 是 HTML 变化很激烈的一个版本，目前确定加入的有数式、体裁档、文字自动绕图。其他的如巨集、国际化等都还在发展阶段。但无庸置疑的是，HTML 正朝着一个完整的版面排置及用户界面迈进。

由于 HTML 大受欢迎，需求（或是说压力）来自各个方面的。很多新的功能及原有功能的扩充都试图加入新的 HTML 标准中。HTML3.0 的工作小组为整个 HTML 的结构定了几个目标：

Lingua France for the Web

由于原文并不好翻译，我将之引出。但其意义是 HTML 意图做为一个有包容性的语言，它允许过去在网络上各种不同的信息被表示出来。如 FTP、NEWS、GOPHER…，各种不同的服务都可以用它来表示。用户只需使用一个单一的 HTML 浏览器（browser）就可以看到所有的信息。

简单性（Simplicity）

HTML 应该是一种结构上很简单的语言，它用最简单的形式把所有的信息表达出来。虽然随着功能的扩充，HTML 不再象 1.0 时那样简单，但新加入的功能都以一种很有条理的形式加入，如果新的功能有不同实现的形式，HTML 宁可牺牲一些功能而加强其简单性。如在考虑表格时，HTML3.0 舍弃了传统 SGML 的 CAL 模型而采用了目前各位看到简单的形式。

可塑性（scaleablity）

不同的情况下会有不同的需求。对于许多简单的应用而言，太复杂的结构反而是信息流通的一个阻碍，但在有些情况下却又需要一些比较精确的控制。例如对一个实验的数据资料而言，只需要很简单的形式即可，而当我们要写一本电子书时对文本中每一个元素的要求就很高了。可塑性在某些方面来说可以通过链接、CGI 界面和内建如 JAVA 之类的语言来完成。HTML 3.0 中提供的体裁档（style sheet）就是一个很好的例子，它将文字属性和文字本身分开，这使得用户可以很容易地选择是否要使用它。

平台独立性（Plateform Independence）

无论用什么机器，使用什么样的配备，都可以得到可以接受的结果，即使我们使用一个文字模式的浏览器（browser）也可以得到一定程度的效果。即使象表格及数式等很难用文字表达的元素也不例外。

2.2 HTML的文本组织

相信再早一年，可能几乎所有的人都不知道什么是WWW，更不会有人知道什么是HTML。但在短短的一年内WWW几乎席卷了整个网络，即使不知道Internet，不知道UNIX的人也都知道有一个叫Web的东西，我们可以从中查到很多很多的数据。类似的工具其实已经有很多个了，为什么WWW会这么红呢？

也许我由HTML3.0的文件中引一段话出来

HTML markup can represent hypertext news, mail, documentation, and hypermedia; menus of options; database query results; simple structured documents with in-lined graphics; and hypertext views of existing bodies of information.

所以知道HTML是被设计成可以用来包装任何形式信息的工具。经过长久的发展，在电脑上我们已经积累了难以估计数量的信息，现代科技的发展使得整个世界每天都会产生一个人一生也读不完的信息。有很多人用很多不同的方法制造出很多不同的信息，你必须学会用各种不同的程序，甚至不同的电脑才可能得到每一个信息。如何消化信息本身已经令人吃不消，如果还要同时去学如何用不同的程序去读这些信息，那简直令人发狂了。

就看一下前面引文中提到的几个你就知道其中的严重性了，你必须学会用tin或BBS以便读newsgroup。你必须学习mail以便和别人交换信件。你必须学会用Ghsoscript以便阅读别人给你的Postscript文本文件。为了看一段MPEG格式的动态影象，可能只是短短一二秒，你却必须学习如何用VMPEG或其他的程序来看，甚至为了有比较好的效果你必须进入MS-Windows中去执行一个硬件的MPEG观看器。为了从数据库中拿一点数据，你必须学习如何操作一个象DBASE般复杂的程序。光这些就可以叫人发疯了，更何况这可能只是一个开始……。

WWW让我们把所有的功能放在一个程序中，它让我们用HTML包装所有的信息。让你可以用同样的方法来看所有的信息，同样按一个ENTER，你可能送出一封信，你可能启动一个MPEG动态影象的播放程序。其实在用户的角度来看，不同数据的格式根本不重要．重要的是文本的内容，尤其是用户最后实际看到的结果。通常很少人会在意数据是从BBS或是GOPHER上来的，它们关心的是信息本身，所以我们不应该让用户看到信息是如何存放的。这就是面向文本的意义，在MS-Windows中的OLE就是这种想法的例子。一个程序只要支持OLE，它就可以把任何信息放在其中，虽然实际的数据还是由好几个程序处理，但是对用户而言好象在处理一个程序一样。

一个HTML文本可以看成是很多不同的信息放在一个文件中，它们在文本中各自成为一个或一串命令，HTML语言提供一些方法让数据的提供者决定用户所看到的数据的格式。例如在某台机器中一个子目录中的文件可能被格式化成如图2-1的格式。

你要在WWW中FTP一个文件不再是在命令行上打入一大串的命令，而是简单的用鼠标在文件名称上点一下就可以了。所以HTML看起来好象就是一个GUI设计语言，这是什么意思呢？一个GUI设计语言顾名思义是用来为某一个应用设计一个用户界面的语言，就好象在MS-Windows中的资源文件一样。系统只要看到资源文件就可以知道整个程序界面应该长成什么样子，以前这个资源文件是用手写的，但现在多用一些交互式的用户界面编辑程序来编写。例如Borland的Resource Workshop就是这样的一个程序。